COMMODORE USERS

48355
K

Ahoy!

$2.50/CAN.$2.75 JUNE 1984

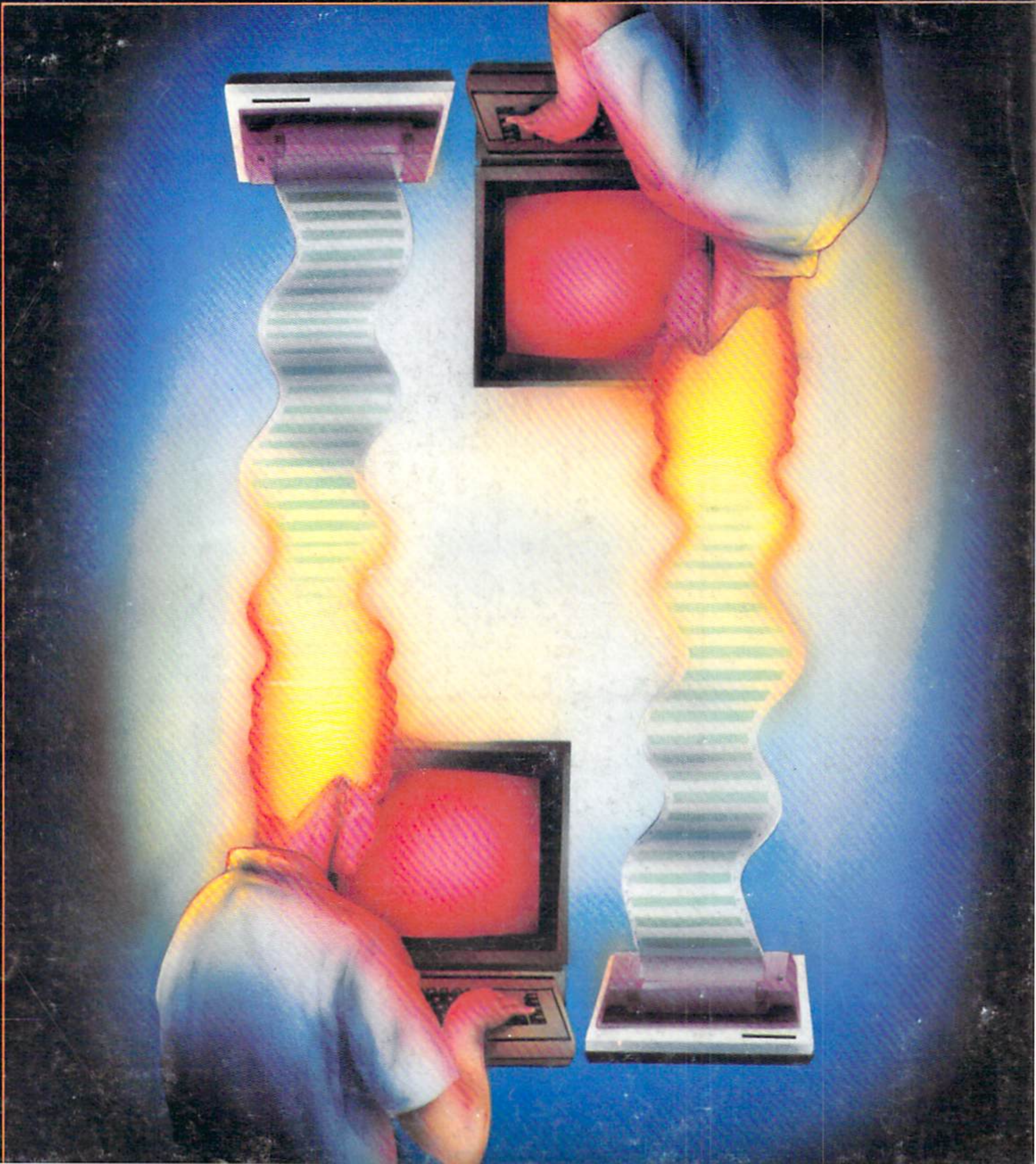
CAN'T PROGRAM? IT'S ALL IN YOUR HEAD!

LEARN TO **CREATE YOUR OWN GAMES** -- FIRST OF A MONTHLY SERIES!

POWER UP WITH **PROGRAM GENERATORS** -- AND WRITE WITHOUT BASIC!

MEMORY MANAGEMENT ADDS BYTE TO YOUR VIC OR 64!

SOUND CONCEPT MAKES VIC SOUND PROGRAMMING A ONE-NOTE AFFAIR!



06
0 71896 48355

"Commodore-ready", . . . and ready for you NOW!

"Cardcorder" DC/1, Data Cassette Recorder/Player

Introducing the "CARDCORDER", Model DC/1, the Computer Cassette that is "Commodore-ready," designed for storage and retrieval of computer data efficiently, economically; with consistent performance. Yet, this fine CARDCO product is priced lower than any similar product with special quality features.

Includes standard connector which is "Commodore-ready"; LED "save" indicator light which confirms data recording on to the tape; handles up to 120 minutes (60 minutes on each side) of any standard tape including existing pre-recorded commercial as well as personal data tapes intended for use with Commodore Personal Computers; ready to go . . . just plug it in and record efficiently.

CARDCO's "CARDCORDER" COMPUTER CASSETTE is a quality data cassette recorder/player in an attractive polystyrene case, with all the standard cassette functions: record . . . play . . . rewind . . . fast forward . . . stop and eject . . . pause. A solid-state designed product of the finest components with auto-stop.

The "CARDCORDER" DC/1 carries a 90 day warranty to original owners.

All CARDCO products are available at your local dealers.



cardco, inc.

313 Mathewson Wichita, Kansas 67214 (316) 267-6525

"The world's largest manufacturer of Commodore accessories."

Commodore™ is a registered trademark of Commodore Business Systems, Inc.

Reader Service No. 268

Ahoy!

CONTENTS

DEPARTMENTS

FEATURES

PROGRAMS

Cover Illustration By: Perry A. Realo

Publisher
Michael Schneider

Editor
Steve Springer

Managing Editor
Robert J. Sodaro

Senior Editor
Tim Moriarty

Consulting Editors
Ben Bova
Morton Kevelson
Dale Rupert

Art Director
Joan Mazzeo-Weinshank

Associate Art Director
Raoul Tenazas

Assistant Art Director
Jo Ann Case

Art Production
Pauline Giordani
Eve Griffin
Peter Neiburg

Technical Advisor
Edward D. Laufer

Circulation Director
W. Charles Squires

Advertising Director
Lynne Dominick

Director of Promotion
Joyce K. Fuchs

Comptroller
Dan Tunick

Managing Director
Richard Stevens

Advertising Representative
JE Publishers Representative
6855 Santa Monica Blvd., Suite 200
Los Angeles, CA 90038
(213)467-2266
Boston (617)437-7628
Dallas (214)660-2253
New York (212)724-7767
Chicago (312)445-2489
Denver (303)595-4331
San Francisco (415)864-3252

AHOY! No. 6, June 1984. Published monthly by Ion International Inc., 45 W. 34th St., Suite 407, New York, NY, 10001. Subscription rate: 12 issues for $19.95, 24 issues for $37.95 (Canada $26.95 and $49.95 respectively). Application to mail at 2nd Class postage rates is pending at New York, NY and additional mailing offices.

No responsibility can be accepted for unsolicited material. All editorial and subscription inquiries and software and hardware to be reviewed should be sent to 45 W. 34th St., Suite 407, New York, NY 10001.

VIEW FROM THE BRIDGE

We're not suggesting that computer programs will ever spring from your head as easily as they do from that of the programmer on our cover. But the articles in this issue dedicated to writing your own programs may help you come closer than ever before.

Debuting this month is our column on *Creating Your Own Games for the VIC and 64.* Orson Scott Card, former Senior Editor of *COMPUTE!* Books, will henceforth appear in every issue of *Ahoy!*, covering the rudiments of generating custom characters, backgrounds, movement, word games, role-playing games, text games, and much more. (Turn to page 13.)

You've heard that *Program Generators* make writing a computer program easier than painting by numbers. But do you know how? Joe Rotello walks you through an actual session with a generator, and reviews two of the available programs. (Turn to page 23.)

And once your appetite is whet, Bernhardt Hurwood will cover several of the software options available in *Code Generating Programs.* (Turn to page 41.)

Of course, old-fashioned blood-and-guts programming is still alive in these pages. But A.J. Kwitowski has made one aspect of it a little simpler with *Sound Concept for the VIC 20.* His program allows VIC users to incorporate sound into their programs without drowning in a sea of loops and POKES. (Turn to page 37.)

Last month, Morton (the K) Kevelson began showing you how to boost the memory of your VIC or 64 to dizzying heights. He finishes the job this issue in *Memory Management on the VIC and 64, Part II.* (Turn to page 19.)

We recommend reading all of the above *before* punching in Bob Lloret's two new game offerings. They're so much fun, you may never get to anything else! *Alpiner* for the C-64 is a downhill ski run so realistic that you'll want to play it with a parka on. Our thanks to Michael Kleinert for a programming assist on this one. (Turn to page 47.)

Continuing in a sporting vein, Bob's *Post Time* features what we feel are the lushest graphics ever to appear in *Ahoy!* Robert Alonso, Technical Editor of *Computer Fun* and frequent *Ahoy!* contributor, translated Bob's 64 game into VIC format. (Turn to page 35.)

What could be more valuable to a Commodore user than a clear understanding of the four main methods of inputting data? Who could provide it better than Dale Rupert? *Ready...Let...Get...Input!* is the subject of this month's *Rupert Report.* (Turn to page 59.)

Dale's *Commodares* are as puzzling as ever. Answer this month's challenge correctly, and you could see *your* name in a future column! (Turn to page 63.)

By now, we don't have to tout Richard Herring's *Educational Software* series; you know it's the definitive guide to choosing educational programs for your children. *Part IV* appears in this issue. (Turn to page 43.)

In response to the many reader requests for more "high-tech" in our *Reviews* section, this issue features *Multiplan, Green Arrow, Magic Desk, Simulated Computer II,* and the *Smart 64 Terminal!* Of course, we've also reviewed the usual gaggle of games, including *Castle Wolfenstein, Starcross, Monster Smash, Spare Change,* and *Brain Strainers.* (Turn to page 49.)

This issue marks our half-year anniversary. It seems like just yesterday that we were finding our magazine grouped with the boating section at newsstands all over New York. Now the sight we see most often is an empty rack where a stack of *Ahoy!'s* had been. Unless they're all selling to boat enthusiasts, we take that as a positive sign.

Thanks for shipping with us this far. The best is still ahead for you, your Commodore, and *Ahoy!*

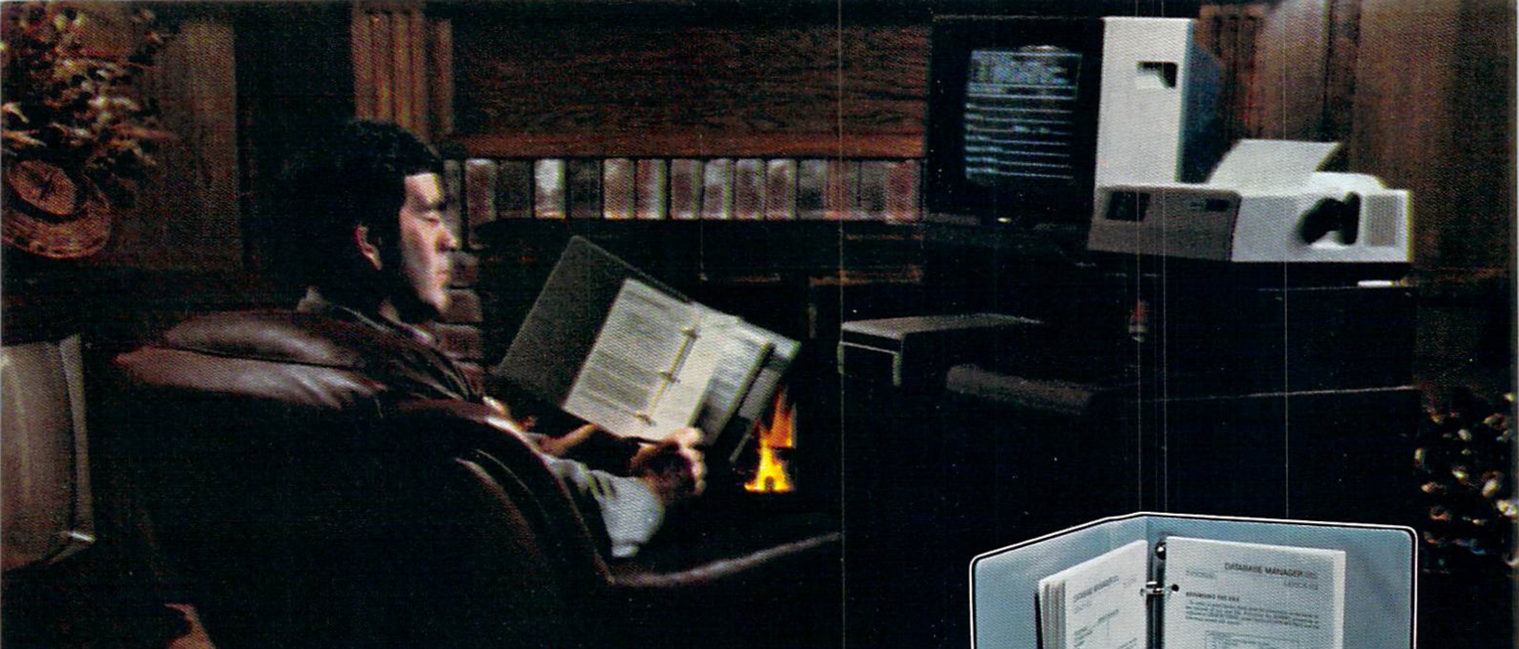
—Steve Springer

BUG REPELLENT CORRECTION!

A defect in our *Bug Repellent* programs for the C-64 (April and May) and VIC 20 (May) prevent them from operating to their full potential. They will catch many, but not all, errors. Please update the *Bug Repellents* you have on file to match those printed on pages 66 and 67 of this issue. The earlier versions are invalid, as this updated version will generate different line codes.

Also in our May issue, two program listings were omitted from *The Rupert Report.* See pages 84-85 of this issue.

Finally, a problem that's not our fault. We received a number of calls from readers having trouble with line 701 of *Lunar Lander* in our April issue. If you look carefully, you'll spot a period 11 spaces from what might have looked like the end of the line.



Commodore 64™ Owners, Relax...



with Mirage Concepts software

Mirage Concepts has mastered the art of uncomplicating software. Before you buy—we help you determine which Mirage Concepts package will meet your need. No guesswork! With your purchase comes a menu-driven program ranked by independent evaluators nationwide as among the finest available. Relax as you learn how to operate your program with clear, concise tutorials written by professional writers... not programmers. For consultation on your special questions, technical support personnel are standing by on a toll-free basis.

For Brochures, Support and Information, Call...
(800) 641-1441
In California, Call...
(800) 641-1442

DATABASE MANAGER, $89.95
• 100% Machine Language • Free Form Design • Sort On Any Field • Calculated Fields • Interfaces to W.P. • Record Size = 2,000 Characters

ADVANCED REPORT GENERATOR, $49.95
• Companion to Database • Totals and Subtotals • Field Matching • Expanded Reports • Sorting (Up & Down) • Calculated Fields

WORD PROCESSOR, Professional Version **$89.95**
• 80 Col w/o Addt'l Hdwr • 100% Machine Language • Spelling Checker (30,000 Words) • Over 70 Single Keystroke Commands • Printer Command File • Interfaces to Database

WORD PROCESSOR, Personal Version **$39.95**
• 100% Machine Language • True Word Wrap • Printed page/line/character counters • Right Justify, Center • Printer Command File • Interfaces to Database

MIRAGE CONCEPTS

MIRAGE CONCEPTS, INC.
2519 W. Shaw Ave., #106 • Fresno, CA 93711
TM—Commodore 64 is a Registered Trade Mark of Commodore Electronics, Ltd.
Reader Service No. 272



Hayden's exciting new handbooks give you the answers:

- *step-by-step instructions so even first-time users gain skills and confidence quickly.*
- *easy access to the information you need.*
- *fascinating games for hours of action-packed thrills so you can have fun while you learn.*
- *practical programs you can put to use at home or in the office.*

Basic Commodore 64™ BASIC
(Coan) Simple, direct approach: Start with short programs. Add a capability. Watch as the desired effect is achieved. Emphasizes the use of the screen editor, immediate mode execution, and memory.
#6456, $14.95

BASIC Commodore 64™ Programs for the Home
(Sternberg) More than 40 useful, money-saving programs for every area of home management: managing finances, arranging schedules, organizing daily chores, and more. Includes BASIC listings, a symbol table, sample data, and at least one output sample for each program.
#5176, $14.95

The Commodore 64™ Illustrated
(Nadler) A unique picture-book approach that illustrates every operational step–from setting up the machine to novice programming. Features a series of simple exercises.
#6453, $10.95

Dr. Aron's Guide to the Care, Feeding and Training of Your Commodore 64™
(Aron & Aron) Practical, helpful guidance in operating and programming your computer, plus complete explanations of every new term and procedure you'll encounter. Includes scores of fun practice exercises with each chapter.
#6450, $12.95

Stimulating Simulations for the Commodore 64™
(Engel) Twelve fantastic, new game programs guaranteed to give you hours of chills and challenges. Test your ability to fight forest fires, search for lost treasures, pilot a space ship, and more!
#5201, $7.50

Mail to:
Hayden Book Company • Dept. AY64
10 Mulholland Drive
Hasbrouck Heights, NJ 07604

Order by Phone 1-800-631-0856
operator AY64 • In NJ (201) 393-6315

Please send me the book(s) indicated below by code number. If I am not completely satisfied I may return the book(s) undamaged, within 10 days for a complete refund. I am enclosing $2.00 to cover postage & handling.
☐ Enclosed is my check or money order
☐ Bill my ☐ Visa ☐ MasterCard

Name

Address

City

State/Zip

Visa/MasterCard # Exp.

Signature

Residents of NJ and CA must add sales tax. Prices subject to change.

HAYDEN

Commodore 64 is a trademark of Commodore Business Machines Inc., which is not affiliated with Hayden Book Company.

SCUTTLEBUTT

NEW COMMODORE BUSINESS COMPUTERS • LIFE INSURANCE AND RETIREMENT PLANNERS • COMPUTERIZED ROAD ATLAS • TERMINAL PROGRAM • HANDICAPPING AND ODDSMAKING PROGRAMS • PATTERN MAKER • HOME MANAGEMENT PROGRAM • LATEST GAME RELEASES

THE CBM-PC?

In a move that propels Commodore Business Machines from neutral waters directly into the PC war zone, the company has unveiled a pair of business computers dsigned to compete directly with IBM's line of personal computers. The announcement came April 4 as the computers were displayed, in prototype form, at the Hanover Fair in West Germany.

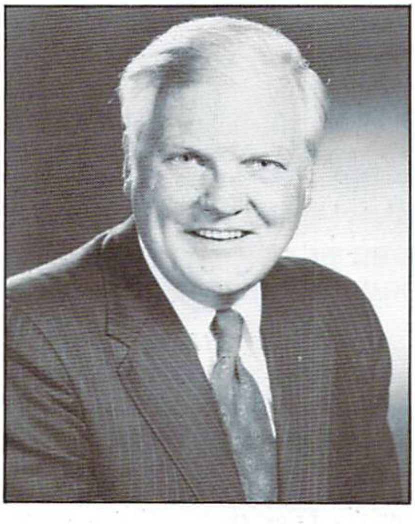
One of the new machines is a 20-pound portable that physically resembles the Hyperion, an IBM-compatible PC. It utilizes Intel's 8088 chip, giving the system 256,000 bytes of RAM. The other new computer offers the same RAM, but is based on the Zilog Z-8000 chip and utilizes the UNIX operating system developed by Bell Laboratories.

Commodore was able to access this technology through licensing agreements with Intel (producer of the 8088 microprocessor chip that powers the IBM PC) and Bytec-Comterm (manufacturer of the Hyperion).

To date, Commodore has been less than successful with their entries into the higher-priced computer market, such as the BX500, B128, and P128. But the publicity garnered by Commodore through its overnight cornering of the lower-end computer market, combined in the case of the 8088-based machine with the appeal of IBM compatibility, could mean a business market breakthrough for Commodore at last.

Also displayed at the Hanover Fair was the previously announced 264. Officials stated that they would ship the computer sometime during the second half of the year. No availability date was given for the other two machines, which analysts interpreted as meaning that the machines would not be available until 1985.

No mention was made of the V364, but a corporate spokesman assured *Ahoy!* that Commodore was committed to the C-64 and VIC 20. "Apple makes a Macintosh every 26 seconds, IBM makes a PC every 16 seconds,"



Marshall F. Smith, the recently appointed president and chief executive officer of Commodore.

said the spokesman. "But Commodore makes a 64 every 5 seconds." Figures were not available for the VIC, but he told *Ahoy!* that Commodore would keep producing VIC's "as long as they're selling." The 264, he added, would probably undergo a name change, so as to avoid confusion with the 64.

WHERE'S THE ACTION?

Speed Handicapper offers C-64 and VIC 20 (+8K) owners a specialized approach to picking a winner. Horse race fans unfamiliar with speed handicapping, a popular handicapping strategy, will find the program useful as an introduction to its fundamentals; experienced speed handicappers will be able to improve their accuracy and lessen the tedium involved.

$29.95 postpaid from High Desert Publishing Company, P.O. Box 36556, Albuquerque, New Mexico 87176.

If you'd rather be on the receiving end of the betting, *The Oddsmaker* will accept wagers, calculate odds and payouts, total betting pools, allow you to take a house cut, and print tickets. It covers all varieties of sporting events, allowing for saving, deleting, and resorting of up to 14 betting events. For the C-64.

$44.95 from CZ Software, 358 Forest Road, South Yarmouth, MA 02664.

THE LONG AND SCROLLING ROAD

For 64 users who want to travel cross-country, but can't read their Rand McNally maps under all the pencil lines, *Roadsearch* has moved vacation planning into the computer age. The program will compute the shortest practical route between any two of the 406 cities/road intersections in its database. It can also determine a route that will avoid toll or other roads, or a route that is longer but more tailored to the driver's

Let The SMART 64 Terminal

COMMODORE 64*

Do The DRIVING



No matter which direction you wish to travel in, experience the advantage of computer communications with The SMART 64 Terminal. Discover the program that puts you on the Right Road to: Public-Access Networks, University Systems, Private Company Computers and Financial Services.

The SMART 64 Terminal designed with Quality-Bred features, Affordable Pricing . . . And Service.

So why not travel the communications highways the SMART way!

Suggested **$49.95** Retail

Accessories included:

- ☐ *Selective Storage of Received Data.*
- ☐ *Alarm Timer.*
- ☐ *40 or 80 Col. Operation*.*
- ☐ *Auto-Dial.*
- ☐ *User-Defined Function Keys, Screen Colors, Printer and Modem Setting.*
- ☐ *Screen Print.*
- ☐ *Disk Wedge Built-In!*
- ☐ *Formatted Lines.*
- ☐ *Review, Rearrange, Print Files.*
- ☐ *Sends/Receives Programs and Files of ANY SIZE.*
- ☐ *Adjustable transmit/receive tables allow custom requirements. These and other features make The SMART 64 Terminal the best choice for grand touring telecommunications.*

*Commodore 64 registered trademark of Commodore Business Machines Inc.
*Supports 80 column cartridge by Data 20 Corporation

Dealer Availability
Call (203) 389-8383

MICROTECHNIC SOLUTIONS
P.O. BOX 2940, NEW HAVEN, CONN. 06515

Reader Service No. 279

TOTL SOFTWARE ROLLS OUT ANOTHER WINNER!

DataBase Management for the Commodore 64™
TOTL.INFOMASTER 3.6—only $50 on disk



Money-Saving Bonus Paks of 64 Software

(BP-1)—(disk)
totl.text/ totl.speller/totl.label
reg. price $103 NOW **$79**

(BP-2)—(disk)
totl.business/ totl time manager/ totl.infomaster/totl.text
reg. price $228 NOW **$159**

(BP-3)—(disk)
totl.infomaster/ totl.text/totl.speller
reg. price $129 NOW **$99**

(BP-4)—(disk)
totl.text/ totl.speller/ research assistant
reg. price $118 NOW **$89**

(BP-5)—(tape)
totl.text/totl.label
reg. price $60 NOW **$49**

Commodore 64 and VIC 20 are trademarks of Commodore Business Machines Inc.

INFORMATION AND ORDER COUPON

	TAPE	DISK
TOTL.TEXT 2.0 (VIC + 8K)	☐ 24.95	☐ 28.95
TOTL.TEXT 2.5 (VIC + 16K)	☐ 34.95	☐ 38.95
TOTL.LABEL 2.1 (VIC + 16K)	☐ 19.95	☐ 23.95
TOTL TIME MGR. 2.1 (VIC + 8K)	☐ 29.95	☐ 33.95
RESEARCH ASST. 2.0 (VIC + 8K)	☐ 29.95	☐ 33.95
TOTL.BUSINESS 3.0 (VIC + 24K)		☐ 84.95
TOTL.TEXT 2.6 (C-64)	☐ 39.95	☐ 43.95
TOTL.SPELLER 3.6 (C64)		☐ 34.95
TOTL.LABEL 2.6 (C-64)	☐ 19.95	☐ 23.95
TOTL TIME MGR. 2.6 (C-64)	☐ 34.95	☐ 38.95
RESEARCH ASST. 2.0 (C-64)	☐ 34.95	☐ 38.95
TOTL.INFOMASTER 3.6 (C-64)		☐ 49.95
TOTL.BUSINESS 3.6 (C-64)		☐ 94.95
BONUS PAK #________		________

Check, Money Order or C.O.D.* also accepted.
*C.O.D. orders $2.00 additional (CA residents add 6½% sales tax)

Total ________
C.O.D. Charges/Sales Tax ________
Shipping & Handling $3.00
Amount Enclosed ________

FOR ORDERING ONLY—CALL OUR TOLL FREE NUMBERS
Continental U.S. 1-800-351-1555, California 1-800-351-1551
Hawaii and Alaska 415-943-7877

☐ SEND MORE INFORMATION (no charge for catalog)

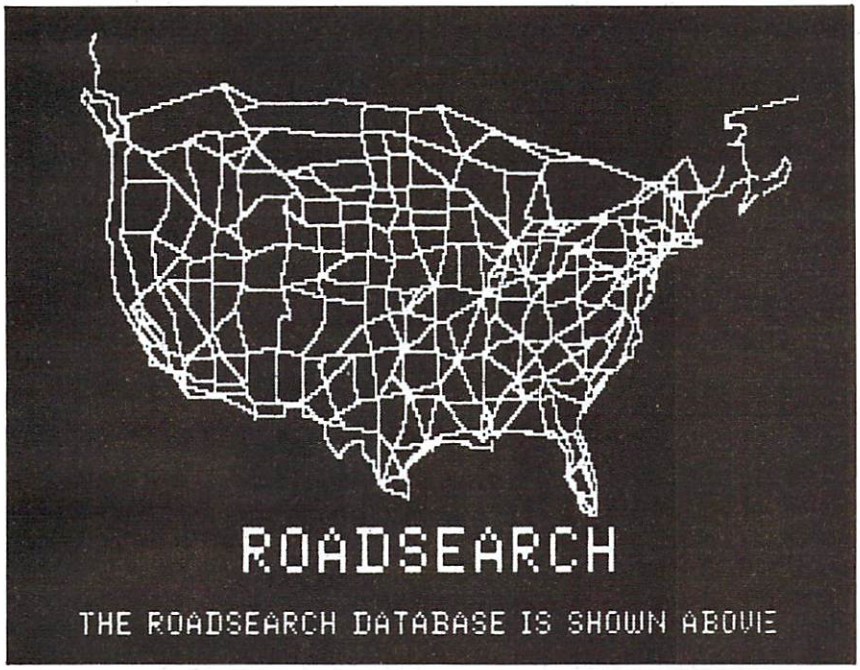
Name ________
Street ________
City ________ State ________ Zip ________
Phone ( ) ________ ☐ MC ☐ VISA
Card # ________ Exp. Date ________

TOTL SOFTWARE, INC.
quality you can afford
1555 Third Avenue
Walnut Creek, CA 94596
415/943-7877

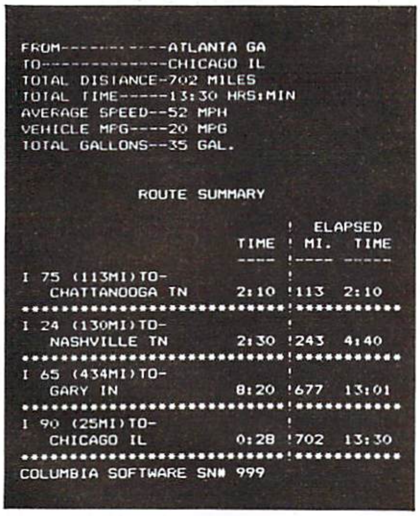
Other VIC 20™ and Commodore 64™ software available from your dealer or directly from TOTL Software:
Word Processing (totl.text)
Spelling Checker (totl.speller)
Mailing List & Label (totl.label)
Business Accounting (totl.business)
Time Management (totl time manager)
Keyword Cross Reference (research assistant)

TOTL Offers You...
low prices and high quality • 30 day money-back guarantee on direct purchases • an interactive family of software • product registration • customer support • free informative newsletter • regular upgrades at reduced cost • availability in many stores • 800 numbers for ordering convenience • prompt shipment of direct orders • savings coupons with each order • money-saving bonus paks • two years and 45,000 products strong

Reader Service No. 280



Roadsearch ***provides routes, mileage, travel times, and fuel usage.***
READER SERVICE NO. 284



406 cities in its database.

needs. Printed outputs include driving route, distances, travel time, and fuel usage.

Roadsearch Plus adds a roadmap development system that allows users to customize their own roadmaps, adding up to 50 towns and 100 road segments anywhere in North America, including local roads, shortcuts, and new destinations.

Roadsearch ($34.95) and *Roadsearch Plus* ($74.95) are both available on disk from Columbia Software, P.O. Box 2235C, 5461 Marsh Hawk, Columbia, MD.

CABLE NEWS

Before you turn your electronic typewriter into an expensive paperweight, consider turning it into a printer for your computer. SuperCord II, an update of the original interface from Cord Ltd., contains a 4K memory buffer, allowing a typewriter to print data at its own rate of speed (usually 110 BAUD), even if data comes from the computer faster.

The Commodore 64, VIC 20, and 8032 PET are among the computers that SuperCord II will link with such electronic typewriters as Adler, Brother, Royal, Smith-Corona, and Silver-Reed. (Take note: that's electronic, not electric, typewriters.)

Cord Ltd., 1548 Brookhollow Drive, Santa Ana, CA 92705.

You can't fit a square peg into a round hole—nor can you fit the 8-pin din plug of the cable that comes with Commodore's 1702 monitor into the 5-pin connection found on older models of the 64. The new 5-pin din plug from Bytes & Pieces, though, will allow you to make the connection for $24.95 plus $2.00 handling.

Bytes & Pieces, 550 N. 68th Street, Wauwatosa, WI 53213.

PLANNING AHEAD

Ever try to get an insurance salesman out of your living room? Advanced Financial Planning has provided one that you can simply power down. Actually, *Life Insurance Planning* won't sell you insurance, but will let you determine the amount of insurance you'll need to cover funeral costs, family living expenses, fu-

ture college tuition, and other costs. The program accounts for inflation and present and future sources of income.

On the sunnier side is the same outfit's *Retirement Planning*, which helps you to construct a financial gameplan that accounts for inflation, investment returns, income needs, and the like. It will calculate inflation rate, real income, and savings required to accumulate the desired retirement fund.

Each C-64 program is $29.95, or $49.95 for both (shipping included) from Advanced Financial Planning, 20922 Paseo Olma, El Toro, CA 92630.

ON THE CHARTS

According to Billboard Magazine, these are the top-selling Commodore-compatible programs in the categories of Entertainment, Education, and Home Management for the week ended 4/7/84. Only the Commodore-compatible members of the Top 10 or 20 are listed, hence the many missing ranks.

Entertainment: *Flight Simulator II*, Sublogic (#1); *Zork I*, Infocom (#3); *Beach-Head*, Access (#4); *Pinball Construction Set*, Electronic Arts (#7); *Blue Max*, Synapse (#8); *Archon*, Electronic Arts (#11); *Choplifter*, Broderbund (#12); *Q*Bert*, Parker Brothers (#13); *Deadline*, Infocom (#14); *Sargon II*, Hayden (#17); *Temple of Apshai*, Epyx (#18); *Donkey Kong*, Atari (#19); *Mystery Master: Murder by the Dozen*, CBS Software (#20).

Education: *MasterType*, Scarborough (#1); *Music Construction Set*, Electronic Arts (#2); *Facemaker*, Spinnaker (#3); *KinderComp*, Spinnaker (#4); *Spellicopter*, DesignWare (#5); *In Search of the Most Amazing Thing*, Spinnaker (#6); *Computer Study Program for the SAT*, Barron's Educational Series (#7); *Type Attack*, Sirius (#8); *Snooper Troops I*, Spinnaker (#9); *Early Games*, Counterpoint Software (#10).

Home Management: *The Home Accountant*, Continental (#1); *The Tax Advantage*, Continental (#2); *Homeword*, Sierra On-Line (#5); *Multiplan*, HesWare (#7); *Paperclip*, Batteries Included (#10).

The above is copyright 1984 by Billboard Publications, Inc., compiled by the Billboard Research Department, and reprinted with permission.

TERMINAL PROGRAM

Two releases from Creative Equipment:

The *Commander Ultra Terminal-64* is an auto-start, self-contained cartridge utility that permits users to communicate with other computers such as the Source, Compuserve, etc. It allows the saving or printer-dumping of material.

Comparing Whole Numbers, a 4-level program, includes instructional materials. It's geared for children 7 to 12 years old.

Creative Equipment, 6864 West Flagler Street, Miami, FL 33144.

THE SCARBOROUGH MENU

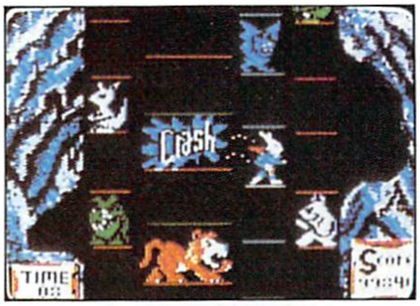
Three from Scarborough:

Phi Beta Filer, a multipurpose home management program, incorporates color and sound and sorts and prints list material alphabetically and numerically. The program's Quiz Master mode can be used to develop games or prepare tests. Ready-to-use files include forms for listing credit cards and home inventories, cataloguing collections and sports records, finding tax deductible expenses, and scheduling family occasions.

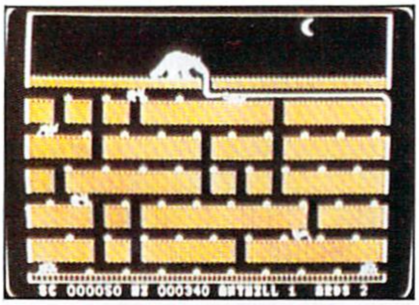
Pattern Maker (scheduled for July release) is an art and mathematics program that allows users to construct, manipulate, and animate symmetrical patterns in color. Forms can be saved and printed.

Run For The Money, a business game by Tom Snyder, shipwrecks you on the planet Simian. There you must develop a business in order to make the money to repair your ship. Players make decisions on how much to pay for materials, what to charge for products, and the like. Slated for June release.

Scarborough Systems, Inc., 25 North Broadway, Tarrytown, NY 10591.



READER SERVICE NO. 250



READER SERVICE NO. 251
Super Bunny *(top) and* **Ardy.**

GO ANIMAL

Two Datamost games newly released on disk for the 64:

As *Ardy the Aardvark,* you'll stick your tongue into an underground maze to lap up ants. But some of the ants will sting, and spiders and other pests will also attack.

If the idea of a mouthful of insects gags you, try the wholesome veggies served up by *Super Bunny.* Here, you'll guide the hare to the carrots that will enable him to beat up his enemies.

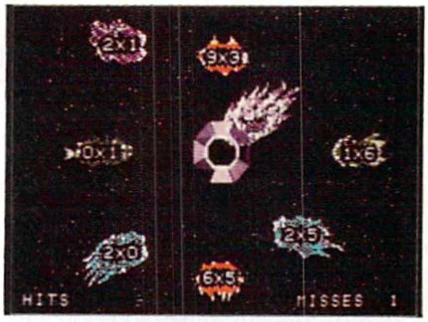
Datamost, 8943 Fullbright Avenue, Chatsworth, CA 91311.

Creative Software has released its first disk-based C-64 game in *DragonHawk,* a battle between an evil flying serpent and a righteous winged thing. At the serpent's service are hordes of flying buzzards, iguanas, phoenix birds, dragon puppies, bats, and mosquitoes. Another hazard is a lightning storm that appears if *DH* remains at a level for too long. He has as many lives as he has feathers, indicated by an on-screen feather count (increased each time he destroys an enemy).

Creative Software, 230 East Caribbean Drive, Sunnyvale, CA 94089.

Described as an "easier graphic adventure game than most," *Unicorn Adventure* sends a young mythical beast north, south, east, and west in search of its mother. Co-designed by a sixth-grader and her mother, the game is geared toward young players. On disk or tape for the 64.

Loggins Enterprises, P.O. Box 10265, Bradenton, FL 34282.

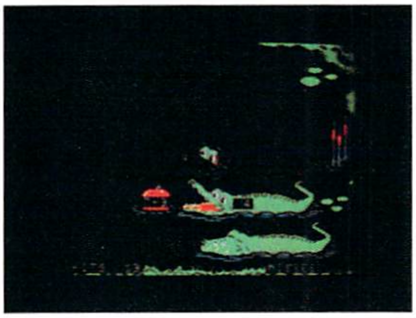


Meteor Multiplication *from DLM.*
READER SERVICE NO. 252

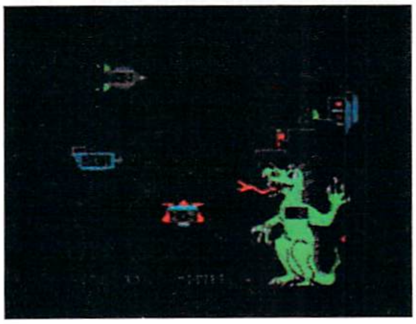
IT'S ARCADEMIC

We seldom dwell on packaging, but DLM's Arcademic Skill Builders in Math have taken us straight back to our youth with their new boxes. Each features a flip-picture—those rectangular glass scenes, once the rage of the 5&10, that change as you tilt them. DLM thereby wins *Ahoy!*'s June 1984 Best Packaging Award, Nostalgia Category, entitling them to a rundown of their Arcademic educational line.

All 12 Arcademic Skill Builders (six math, six language arts) about to be mentioned feature paddle or keyboard control, nine speed variations, and a $34.00 price.



READER SERVICE NO. 240



READER SERVICE NO. 241
Alligator Mix ***(top) and*** **Dragon Mix** ***practice combined skills.***

Alien Addition requires you to fire correct answers *Space Invaders*-style at descending equations. *Minus Mission* poses the same task against blobs of dripping slime. *Alligator Mix* combines addition and subtraction as you feed the gators equation-containing apples. *Meteor Multiplication* has you ward off meteors closing in on your space station. In *Demolition Division,* you fire mortars at advancing tanks. *Dragon Mix* combines multiplication and division as you help a benevolent behemoth incinerate alien invaders with his flaming breath.

Moving to the other side of the brain: *Verb Viper* practices subject agreement with verbs in present, past, and past participle form. *Word Man* sends you tooling through a maze forming consonant-vowel-consonant combinations. *Word Invasion* has you manipulate the parts of speech—nouns, pronouns, etc.—with an alien octopus. In *Spelling Wiz,* you zap misspelled letters into words. In *Word Radar* you match basic sight words from a control tower. And in *Word Master* you identify pairs of antonyms, synonyms, and homonyms.

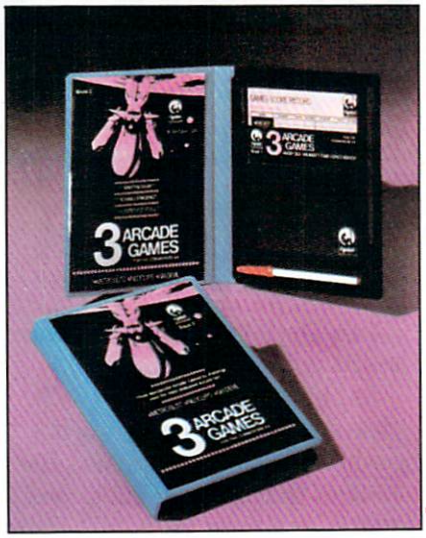
The Arcademic Drill Builders ($44.00 each) incorporate the gameplay of various of the above games, with the added feature of allowing the player to program in his own problems. Up to 36 games can be created and saved on each diskette. Titles are *Wiz Words, Alien Action, Meteor Mission, Alligator Alley, Idea Invasion,* and *Master Match.*

All 18 of the above come on diskette for the C-64.

DLM, One DLM Park, Allen, TX 75002.



Wiz Works: ***user-programmed.***
READER SERVICE NO. 242





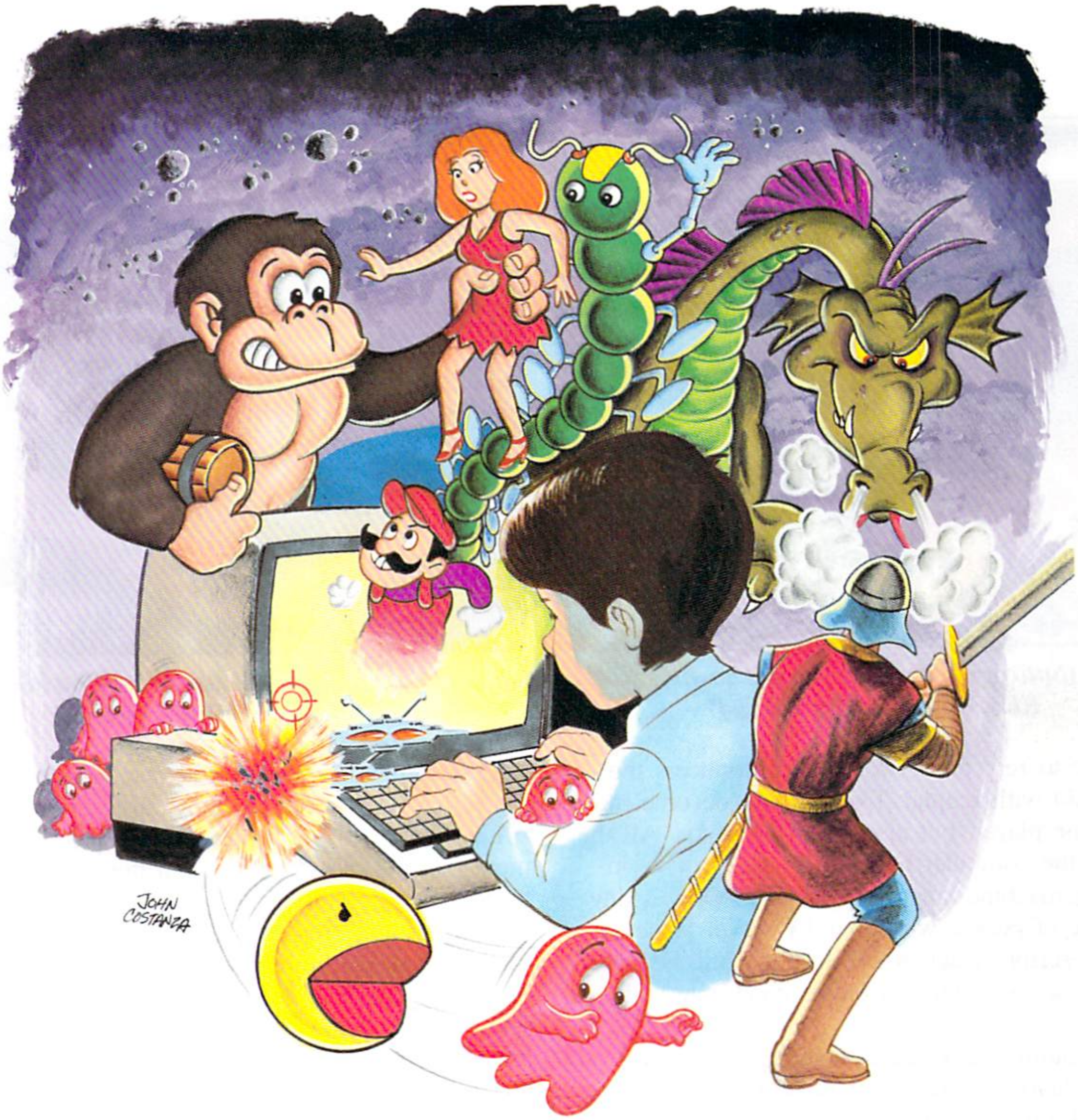
Cymbal's 3-in-1 arcade games package, and 1600-question trivia contest.
READER SERVICE NO. 243

THE GRAPHIC CYMBAL

Cymbal Software's *Arcade Games Book 2,* like its forerunner, features a trio of games in a single package: *Metro Blitz* (defend your domed city from nuclear armageddon), *Ski Devil* (downhill racing, complete with a St. Bernard with brandy flask), and *Neoclyps* (slide-and-shoot with a horizontally-scrolling screen). For those who missed Book 1, it's *Mummy's Tomb (Dungeons & Dragons*-type treasure hunt through a labyrinth), *Moby Dick* (seafaring war game wherein you defend against planes and subs), and *Space Mayhem* (space shoot-'em-up).

Also newly available to U.S. gamers is *Trivia 1,* offering three levels and over 1600 questions. It can be played by individuals or teams, with players setting the point-limit of a particular round.

Cymbal Software, Inc., 250 Don Park #11, Markam, Ontario, Canada L3R 2V1. □



© John Costanza 1984

CREATING YOUR OWN GAMES ON THE VIC AND 64

PART I: FROM THE PLAYER TO THE SCREEN

By Orson Scott Card

The beginning of every game is communication—the player has to be able to tell the computer what to do, and the computer has to be able to tell the player what's happening.

Maybe when you got your Commodore, you were planning to use it for real work—doing budgets, keeping records, helping to educate children, word processing. But if you're like most of us, you also planned on getting a few games. In fact, I'd guess that most home computers are used as much for games as for all other uses combined.

What's more, I don't think that's bad. In fact, I think that's great. Because I believe that games are the best use of the home computer. It's no accident that your VIC or 64 is hooked up to a television set. Most families that don't have computers spend hours a day with the TV on, passively being entertained. But as soon as that computer is hooked up, entertainment isn't passive anymore. When you play videogames, you're still getting all the storytelling that TV normally provides, but now you're part of the story.

Better yet, you can program your own games. You





Joysticks (Suncom TAC-2 shown) put a whole world at your fingertips; trackballs like WICO's add versatility.
READER SERVICE NO. 247 *READER SERVICE NO. 248*

don't have to rely on networks and producers the way you do with regular TV. You can become, not just a game player, but also a game creator. All the power of the computer is there for you to use, in BASIC or machine language—or both. Only a tiny percentage of people who own TV's ever write a television script or act in a TV show. But *everybody* who owns a computer can create good entertainment programs.

This column is devoted to helping you develop ideas and learn techniques of game programming. We'll work on fast action, arcade-type games; word games; adventure games; building block games. Most programming techniques will be explained and demonstrated in BASIC, but of course every BASIC technique can be duplicated in machine language. Along the way, I'll review some software and hardware, discuss some game theory, and sometimes include a complete, playable game—though usually I'll only include enough programming to demonstrate a technique. Every technique will be demonstrated on both the 64 and the VIC—except in those rare cases, like sprites and fancy sound, where the VIC just can't do the same job.

And along the way, I hope you'll have as much fun creating games as I do.

LETTING THE PLAYER HAVE CONTROL

One of the best things about computer games is that the player is in control of events, to one degree or another. What the player does determines what happens on the screen. But the computer has to be told what the player wants to do. The player's control comes through a communications system—the hardware that the player actually plays with, and the way the program receives and responds to the messages that come from the hardware.

Let's start with the hardware, the piece of equipment that the player uses to tell the computer what to do.

Your computer came out of the box with a versatile and powerful communication system built right in: the keyboard. It has 59 regular keys, and each sends a unique message to the computer. Also, there are three shift keys—SHIFT, CTRL, and the Commodore logo key—that can be pressed in eight different combinations, so that in theory you could use the keyboard to send 472 separate messages to the computer!

In comparison, most other controllers might seem pretty limited. A joystick, for instance, can only tell you up, down, left, right, four diagonals, or no movement, and whether the button is pressed or not. That's a total of 18 unique messages. This doesn't mean, however, that the keyboard is "better" than the joystick. It just means that each has different strengths and weaknesses. And as you program your own games, it's a good idea to decide which control system will work best for your game.

So in the first half of this column, we'll discuss each controller and its advantages and disadvantages. Then we'll use the one communication device that *every* VIC and 64 owner has—the keyboard—and do the simplest, most basic game technique of all: moving a figure around on the screen, under the player's control.

THE KEYBOARD

The letters, numbers, symbols, and words printed on the keys tell you what those keys mean when the

operating system or BASIC is controlling the computer. Press the key A and the character A will appear on the screen.

Since the keys follow the pattern of a standard QWERTY typewriter keyboard, this is the most obvious system to use for word games, or text adventures. Touch typists have already memorized the key positions, so they can give the computer instructions without even thinking about where each particular key might be.

But you don't have to use those standard meanings. In fact, action games almost force you to change the meanings of the keys. Fortunately, that's easy to do, and if you use the keyboard carefully, your program can get information from the keyboard faster than you might think possible.

This is because each of the regular keys sends a unique code, a number from 0 to 63, to the computer. The operating system puts that code number at location 197. If no key is pressed, the operating system puts the number 64 at that location.

To see the code that each key generates, try this simple program:

```
10 PRINT PEEK(197):GOTO 10
```

As soon as you RUN this line, the computer will start printing 64 down the lefthand edge of the screen. Now press any key *except* SHIFT, CTRL, RESTORE, SHIFT LOCK, RUN/STOP, and the Commodore logo key. (In particular, notice the number that comes when you press function 5 and function 7—we'll be using those keycodes later.)

Now, take a key you've pressed, and while pressing it, press the SHIFT key. There's no change! SHIFT doesn't make any difference to the number stored at location 197. Now press two regular keys at the same time. Only one keycode is noticed by the computer.

To end the program, press RUN/STOP.

The SHIFT, CTRL, and Commodore logo keys generate a separate keycode, stored at location 653. Try this program:

```
10 PRINT PEEK(653):GOTO 10
```

When you press SHIFT, the computer prints a 1; when you press the Commodore key, it prints a 2; and when you press CTRL, it prints a 4.

Interestingly enough, when you press these keys in combination, it makes a difference. SHIFT and Commodore together make a 3; SHIFT and CTRL make 5; Commodore and CTRL make 6; and all three together make 7. This means that each combination of these keys makes one of eight unique signals to the computer, from 0 (none pressed) to 7 (all pressed). By reading location 653 and location 197 separately, you can get a large number of different codes:

```
10 A=PEEK(197):B=PEEK(653)*100:PR
INT A"  "B"  "A+B:GOTO 10
```

See how many different codes you can get?

The drawback to using the keyboard is that as soon as you stop using the regular QWERTY keyboard pattern, the players of your game have to memorize the meaning of each key combination they're going to use. In practice, this means that instead of hundreds of messages, your players are only going to be willing to learn a few dozen at the most. Also, some key combinations are pretty hard to use. You have to be a contortionist to use SHIFT-CTRL-Commodore together and still press another key at the same time!

Another drawback is that preschoolers don't always know how to read the keyboard—so games that rely on the keyboard usually don't work well for really small children.

Word games, text adventures, and some building block games work best with the keyboard.

JOYSTICKS

Joysticks provide less information and are a bit slower to program, especially on the VIC, which splits joystick information between two separate memory locations. Joysticks, though they provide very little information, are easy to play with. This is especially true with games where pushing the joystick left makes a figure move left, and so on. There's a logical relationship between the direction you push the joystick and what happens on the screen.

Things get a little trickier if you have an odd movement pattern, but at least there are only a few possible choices to make—it's easy to learn what each choice means. Joysticks are also cheap, which is one reason why it's the most-used controller in game programming—chances are pretty good that most computer owners will have a joystick.

Joysticks cause problems for little kids. The concept of joystick movement is easy for them to grasp, but actually moving the joystick is sometimes more than small, undeveloped muscles can handle. Three- and four-year-olds often can't use a joystick at all.

Joysticks work best for fast-action games that pri-

marily use up, down, left, and right movement, like maze games and climbing games. They can also be used to control a cursor on the screen to select items from an onscreen menu.

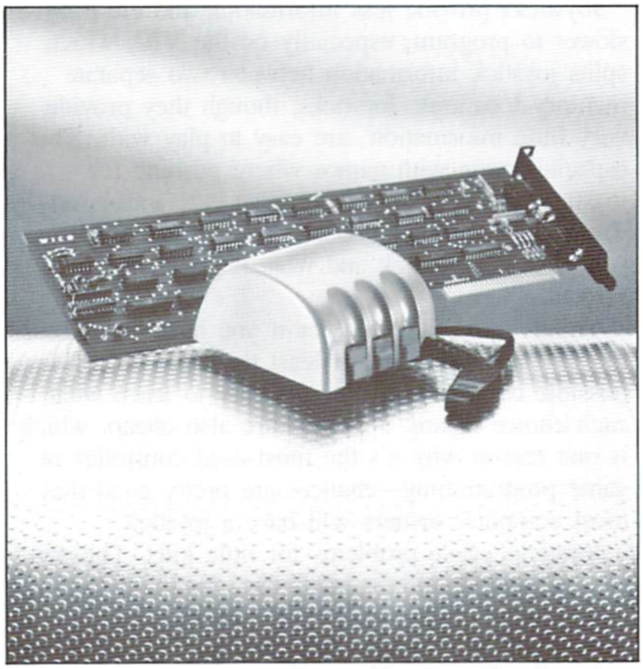
ROLLER CONTROLLERS: TRACK BALLS AND MICE

Like joysticks, a trackball or a mouse (a mouse is just a trackball upside down) only tells *direction* of movement. However, roller controllers can give many more directions than just left and right and up and down. The precision can be excellent, and anybody can use them.

Well, there's an exception. Only people with clean tables can use a mouse. I've never had a clean desk or table in my life. Mice have a way of hiding under piles of urgent projects.

The strength of rollers—quick movement in almost any direction—is also their drawback. They only give direction, not location. If you want to get to a certain spot on the screen, you have to *travel* there—figure out where you are and then get from one place to another. There are also many games in which you want to travel in an exact horizontal line. Imagine trying to play *Breakout* with a trackball.

Rollers work best in games like football, where the player needs to be able to move in any direction, and in games where players move a cursor to select items from an onscreen menu, like the *Pinball Construction Set.*



WICO's mouse (non-Commodore) tells direction. READER SERVICE NO. 276

PADDLES

Paddles are hard to program from BASIC on the VIC and 64—it's almost required that you use a machine language routine to read paddle controllers. Also, since few programs are written for paddles, few VIC and 64 owners have a set. Still, there are some things paddles do better than any other controller.

Paddles can send many possible values to the computer, but the numbers are arranged in a row. That is, you can only move the paddle in two directions, clockwise or counterclockwise, which means that even though you can send more *numbers* to the computer, you really can choose only two directions. However, you can instantly position the paddle at any position along that line. This is much faster than a joystick, which can only tell the program to move in a certain direction, not how *far* to move in that direction.

Paddles work best with *Pong*-type rebounding games, where you are controlling a figure that only moves back and forth in a straight line. For this purpose, paddles work much better than any other controller.

LIGHT PENS

Light pens have the advantage of pointing directly at the screen—the communication between the pen and the graphics display is almost perfect. Also, little kids can use light pens easily—it's as natural as pointing.

I have never personally programmed with a light pen, but I understand it's easy. One drawback is that the TV or monitor has to be in easy reach. Also, the program has to be very fast to cope with movements of the pen. It's easy to move the pen faster than the program can handle. For instance, if you quickly drew the pen from the upper lefthand corner of the screen to the lower righthand corner, you could probably complete the motion in less than a second. The very fastest that the computer can read the light pen is once every sixtieth of a second. So at the very most (and no BASIC program could work this fast), the computer would catch and respond to only sixty different positions of the light pen in a single one-second movement.

Also, the light pen can't do anything *but* point to something on the screen. If you have plenty of space to display a menu of choices, or if the player only has to control a single player-figure, that's not a problem. As soon as you need more sophisticated communication, the light pen has problems.

Light pens are ideal for selecting from an on-

screen menu—you always know exactly where you are. And if your game allows the player's figure to move instantly from any place on the screen to any other, the light pen would work very well. However, when you only want the player to control direction of movement, and not speed, the light pen is actually less effective than the joystick or roller.

TOUCH PADS

Touch pads, like the Chalk Board Power Pad, are rectangular tablets. When you touch the tablet at any point, the pad sends two numbers back to the computer—the horizontal position and vertical position of the point that you're touching. This allows hundreds and hundreds of possible messages to be sent.

In programming, you can use a part of the touch pad to correspond directly with the screen. That is, touching the lower left-hand corner of the pad causes the cursor to move to the lower left-hand corner of the screen. So the correspondence with the screen can be quite exact, just like the light pen.

However, since you don't have to use the entire touch pad for screen control, you can also use sections of the pad for offscreen messages, the way the keyboard does. For instance, the drawing program I use with the Chalk Board Power Pad, *Leo's 'Lectric Paintbrush,* comes with an overlay that sits on top of the touch pad. The bottom two-thirds of the pad is used to function the screen, but the top section contains two rows of hexagons that work like typewriter keys. You press the pad on the hexagon that controls the function you want. For instance, there are color hexagons to choose the color I want to draw with; a background color; hexagons to allow me to save drawings to disk or cassette; and even controls to allow me to save a particular pattern and repeat it anywhere I want on the screen. All of these commands are instantly available, and they don't use up any space on the screen.

Another software package I've used with the Chalk Board Power Pad is *MicroMaestro.* The overlay with this program puts a two-octave piano keyboard at the bottom of the pad, complete with black and white keys. Above that is a musical staff, with the notes drawn on in an ascending chromatic scale. And at the top are hexagons to control things like disk and cassette operations. None of the pad directly controls the screen. Instead, the screen displays the musical staff and the notes in the melody that you play. The program remembers your melody, and when you're playing it back, you can add harmonies to it, which are also remembered.



Light pens like Tech-Sketch's are ideal for children
READER SERVICE NO. 277

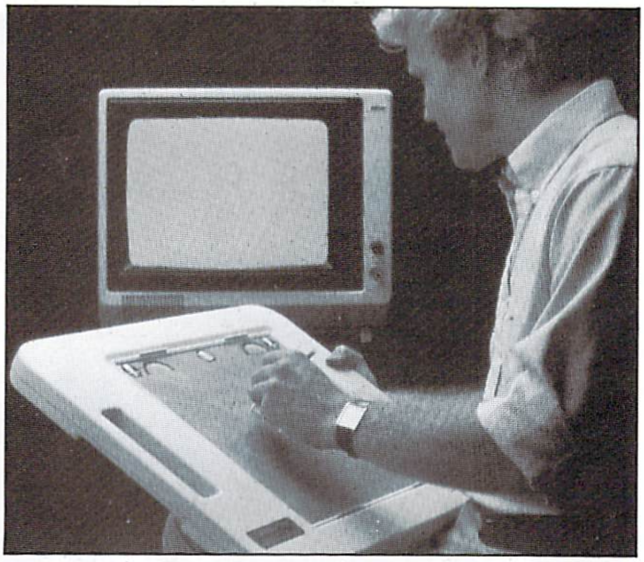
The touch pad is amazingly versatile, especially the Power Pad, with its overlays. You can buy a programmer's kit and use the machine language routines included with it to create your own uses for the pad—and your own overlays.

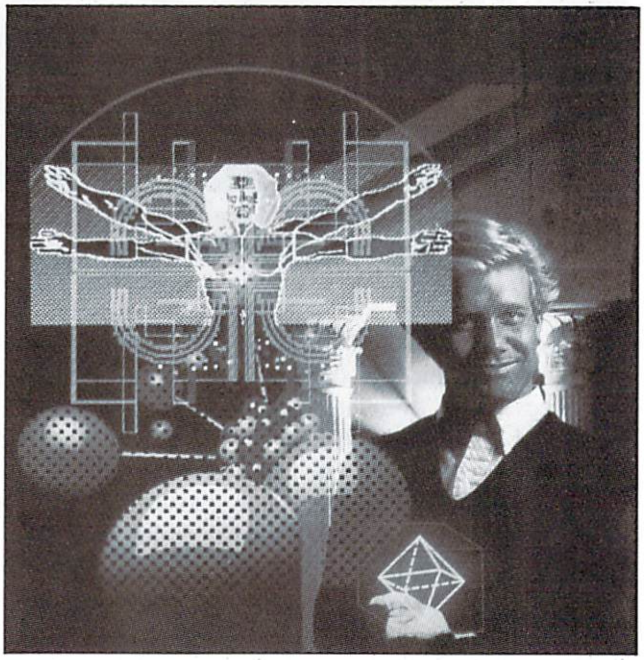
Except for the light pen, touch pads are also the easiest controllers for little children to use. My five-year-old used both the music and drawing programs instantly, and it wasn't long before my three-year-old—who can't use a joystick yet—was drawing and plinking out melodies to her heart's content.

Drawbacks? The price is high, and it takes machine language to use the pad. Also, it's sometimes hard to find your place on the pad. Because you aren't pointing directly to the screen, it's sometimes hard to remember where you were. The pad also wouldn't be a good controller for the kinds of games that rollers and joysticks do best, where the player is controlling the direction in which an onscreen figure moves.

GETTING IN MOTION

The hardware is only one part of the control system in a game. The program itself has as much effect on player control as the controller you use. For instance, if your program only checks for player in-





Touch pads like **Chalk Board** *yield varied graphics.*
READER SERVICE NO. 273

structions twice a second, that means that the fastest the player can move is twice a second. That may sound like speed, until you try it. A player could break a joystick struggling to make the onscreen figure *move.* It feels miserably slow.

A fast-action game has to feel *fast.* However, that doesn't mean you have to use machine language. BASIC is a fast language—you just have to design your game properly. And the most important thing to keep in mind is that the feeling of speed depends almost entirely on how quickly the game responds to the player's control. It must feel to the player as if the onscreen figure responds instantly to every instruction.

With machine language, you would have time to spare—you could keep a dozen computer-controlled figures moving smoothly and play music and calculate the age of the earth without slowing down the player a bit.

With BASIC, you have to be more careful. Every arithmetic operation slows things down a little. In fact, everything that the program does between the times it checks for instructions from the player will make the program *feel* slower to the player.

THE MOVEMENT PROCESS

We'll start by moving a single character around the screen. We'll use the diamond-shaped character, the one you get by pressing SHIFT-Z or using CHR$(122). To make it seem to "move" on the screen, we first erase it, then PRINT it again in an adjacent location. If we want it to move up, we PRINT it in the same column, but on the row above the old location; to move left, we PRINT it in the same row, but in the column to the left.

PRINTing is the fastest way to move around the screen in BASIC. This is because PRINT moves at machine language speeds, while POKEing and PEEKing directly into screen memory is much, much slower. The nicest thing about PRINT is that it handles row and column calculations automatically.

HORIZONTAL POSITION

With the VIC there are 22 legal horizontal positions which we will number from 0 to 21. To PRINT CHR$(122) in position 5, you just use this statement:

```
PRINT TAB(5)CHR$(122)
```

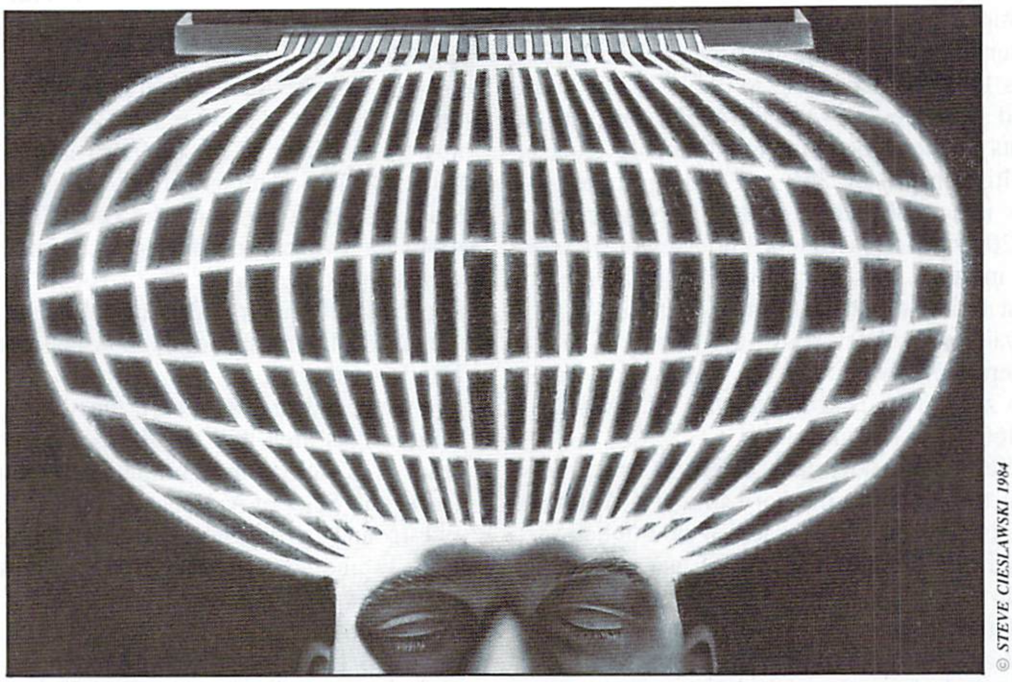
In our program, we'll use the variable PF$ to hold the character CHR$(122). We'll use the variable H to hold the horizontal position. The *old* horizontal position, the one we need to erase, will be held in the variable HX. To move the figure one space to the left, we PRINT a blank at the old location, subtract one from H, and PRINT PF$ at the new location—almost like this:

```
HX=H:H=H-1:PRINT TAB(HX)" ":PRINT
 TAB(H)PF$
```

To give the smoothest possible movement, we do

Continued on page 27

MEMORY MANAGEMENT



ON THE VIC 20 AND C-64

PART II

By Morton A. Kevelson

Last month we discussed the memory maps for the VIC 20 and Commodore 64 as used by the BASIC interpreter. We indicated that both computers have significant sections of RAM which are not used by BASIC. We mentioned that by manipulating certain pointers, we would be able to use this memory for BASIC programs. The pointers, which BASIC uses to keep track of programs and data in memory, are listed in Table I. We also presented two short programs which demonstrated the techniques involved. These programs manipulated the pointers of Table I, allowing both programs to reside simultaneously in memory. Of the two, one was loaded into a part of memory which would not normally be available to BASIC.

THE MEMORY POINTERS

The contents of the pointers in Table I actually represent a specific address in the computer's memory. These pointers are used by the BASIC interpreter to specify the boundary of a particular section of RAM. Each pointer consists of two consecutive bytes which are paired as shown. This allows any of the 65,536 addresses in the computer's memory to be covered. A single byte, consisting of eight bits, can assume only 256 possible values, 0 to 255. This is obviously not enough to represent all the possible memory locations. By storing the address in two parts, the number of possible values is increased by 256 times. The most significant part of the address is stored in the higher memory location of the point-

er pair. To calculate the address value of a pointer, simply multiply the contents of the high byte by 256 and add it to the contents of the low byte. This arrangement is generally referred to as low byte–high byte addressing.

For example, the Start of BASIC pointer tells the BASIC interpreter where the first byte of the BASIC program is located. According to Table I, this pointer is stored in addresses 43 and 44. For the VIC 20, the contents of these locations are 1 and 18 respectively. Multiplying 18 by 256 and adding 1, we get 4609. This is actually where a BASIC program starts in a VIC 20 with at least eight kilobytes of memory expansion installed. Referring to the VIC 20 memory map in last month's issue, note that location 4608 is the first available address above the screen. The BASIC interpreter always sets the contents of this location to zero. This is why the start of BASIC points to 4609 and not to 4608.

The BASIC interpreter manages BASIC programs by dividing the available memory into five distinct areas. The interpreter keeps track of how much memory is in each of these areas by using the pointers of Table I. Each area is actually defined by the contents of two pointers.

The first area is where the BASIC program itself is stored. The start of the program is pointed to by the contents of locations 43 and 44, which we discussed above. The end of the BASIC program is stored in locations 45 and 46 which are actually the pointer to the start of the next area on our list. When a BASIC program is entered from the keyboard, the contents of the Start of Variables pointer are readjusted every time a line is added to the program. In addition, the contents of the next two pointers are adjusted as well. These pointers will also be adjusted when a program is read into the computer, by executing the LOAD command from the keyboard.

Note that the Start of Variables pointer actually marks the first byte after the BASIC program and not the last byte of the program itself. The initial value of the pointer is set two bytes higher than the Start of BASIC pointer. The BASIC interpreter leaves room for two consecutive bytes, containing zeroes, which mark the end of every BASIC program. The first pair of pointers actually delimits the part of memory which will be saved when a SAVE command is executed.

The second area is where the variables data is stored. This area will usually be empty until a program is actually run. The delimiters for this area are set by the Start of Variables pointer in locations 45 and 46 and by the Start of Arrays pointer in locations 47 and 48. Note how each of these pointers actually serves double duty by separating two of the areas used by the BASIC interpreter. As before, the Start of Arrays pointer actually contains the first address after the variable space. Whenever the program assigns a value to a variable, space is allocated in this area for both the variable name and its current value. For dimensioned variables, only the name is stored in this area.

The actual way the data is stored is an interesting topic which I hope to discuss in a future article. The important thing to remember is that as variables are assigned, such as A=52, the Start of Arrays pointer will be increased to mark the first byte available after the variable space. As before, the End of Arrays pointer is incremented as well. These pointers will also be incremented by assigning a variable from the immediate mode at the keyboard.

The third area is where the actual contents of numeric arrays are stored. Whenever a DIM statement is executed, the BASIC interpreter sets aside whatever space is needed for each element in the array. Initially, these will all be equal to zero. The space allocated for the array data is delimited by the Start of Arrays pointer in locations 47 and 48 and by the End of Arrays+1 pointer in locations 49 and 50.

Once a program is loaded and running, the Start of Variables pointer will remain fixed. This is expected, as once the program is in memory its length will not normally be altered by normal execution. The remaining pointers will change whenever the

TABLE 1
BASIC MEMORY MANAGEMENT ADDRESS

MEMORY ADDRESS	FUNCTION	CONTENTS VIC 20	C-64
43 44	Start of BASIC	1 18	1 8
45 46	Start of Variables	3 18	3 8
47 48	Start of Arrays	3 18	3 8
49 50	End of Arrays + 1	3 18	3 8
51 52	Bottom of Strings	0 64	0 160
55 56	End of Memory	0 64	0 160

program assigns a new variable. Changing the value of a previously assigned variable will not affect the values of the pointers. This is because the BASIC interpreter reuses the memory space for numeric variables as they are reassigned. New values simply replace old values as required.

The fourth area is delimited by the End of Arrays+1 pointer in locations 49 and 50 and the Bottom of Strings pointer in locations 51 and 52. This is the empty space that BASIC has left to work with. Whenever a FRE(0) command is executed, the available bytes free is calculated by subtracting the Bottom of Strings address from the End of Arrays+1 address. Try POKEing different values into these pointers and executing the PRINT FRE(0) command. Remember that values greater than 32767 will be displayed as a negative number. The correct answer can be displayed by adding two to the sixteenth power to the result:

```
PRINT2^16+FRE(0)
```

The fifth area is where the BASIC interpreter stores all calculated strings. These are all strings which are not created by a simple assignment statement such as A$="PRESS A KEY". The latter are stored as part of the BASIC program itself. This makes a lot of sense. Why take up additional memory for something which has been explicitly defined in the program? The pointers for this area work differently than for the other areas. The End of Memory pointer in locations 55 and 56 actually represents the start of the string storage area. The Bottom of Strings pointer in 51 and 52 defines the end of string storage. New strings are stored in a top down sequence. Thus the Bottom of Strings pointer and the End of Arrays pointer will always move toward each other. When the two are equal, an out of memory error is generated.

The string storage area is also far more dynamic than the other variable storage areas. When a string variable is redefined, it is stored in a new location without erasing the old value. All this is necessitated by the variable length of strings. This makes it impossible for the BASIC interpreter to assign a specific location for a given string variable. In contrast, numeric variables always occupy the same amount of memory.

All of this string manipulation makes the string storage boundaries rather dynamic. If a string variable is redefined many times in a program, memory can actually be filled by the "discarded" values of the string. To prevent this, the BASIC interpreter has a built-in routine known as garbage collection. When this routine is called, it scans through string storage and throws away all the discarded strings from superseded variable assignments. This action frees up memory for the BASIC program. As a result, the Bottom of Strings pointer may actually move up and down in memory during program execution. This garbage collection routine is what causes the occasional lengthy pauses during the execution of a BASIC program.

MEMORY MANIPULATION

The BASIC interpreter automatically takes care of memory management by continuously keeping track of the pointers. To do this effectively, the interpreter requires that the memory it is working with be in a continuous block. As can be seen from the discussion, there is no room for unassigned memory gaps in the intrepreter's scheme of things. It is for this reason that the three kilobyte memory expander is lost to BASIC on the VIC 20 when eight kilobytes of memory are added. By carefully POKEing different values into the pointers of Table I, we can give the interpreter a hand in keeping track of things. In effect, we will be letting the interpreter know where the additional memory is located and exactly for what it should be used.

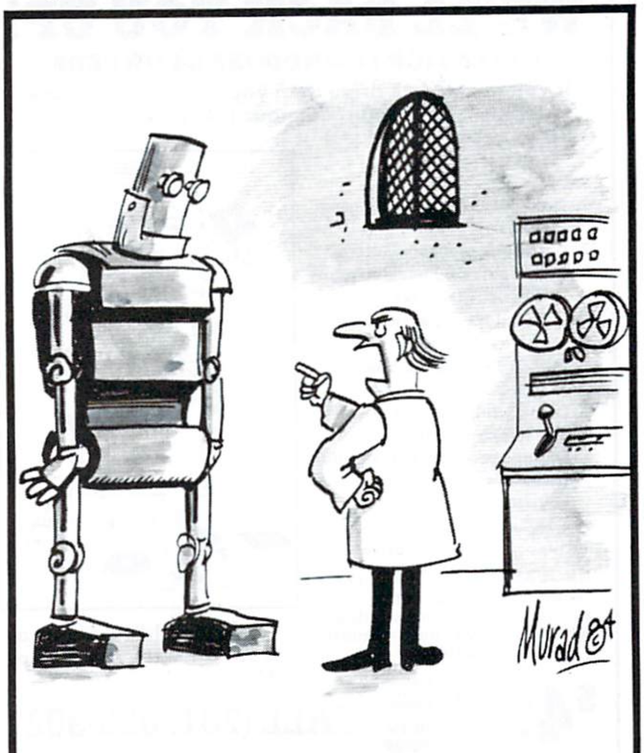
Before we go into the details of the procedure, it will be necessary to understand how some of BASIC's commands affect the pointers.

THE LOAD COMMAND

The results of executing a LOAD command differ considerably between immediate mode and program mode. There is also a difference between tape and disk. When executed in immediate mode, a disk LOAD will reset the pointers and read a program file into memory. The pointers will be set to the start and end of the new program. If a non-relocating machine language LOAD is performed, an out of memory error may result on subsequent commands. This is a result of the pointers in locations 45 through 50 being set above the contents of the Bottom of Strings pointer by the LOAD.

A tape LOAD does not reset the pointers before the start of the read sequence. If the new program is shorter than the old program, the end of the old program will still be in memory. This could give some strange results. For this reason the NEW command should be executed before tape LOADs.

A LOAD under program control behaves somewhat differently. In this case, the pointers are not changed at all when the LOAD is performed. Thus, the pointers will remain set to the values of the original program performing the LOAD. In addition, as soon as the LOAD is complete, the BASIC interpreter resumes execution at the start of the program as pointed to by the Start of BASIC pointer. Finally, when a LOAD command is executed, the interpreter actually reads the BASIC program into memory starting at the location pointed to by the Start of BASIC pointer. This actually forms the basis for the memory management techniques presented here. By carefully changing the Start of BASIC pointer, we can LOAD and RUN a BASIC program from those memory locations not normally accessible to the BASIC interpreter.



"Remember...I made you, and I can break you!"

Two other aspects of the LOAD command are of interest: the relocating LOAD and the non-relocating LOAD mentioned above. The relocating LOAD is the one most users are familiar with. This is the type of LOAD used to read in a BASIC program. For the Commodore 64 and the VIC 20, a very useful feature has been added to the relocating LOAD: the ability to relocate BASIC programs in memory. This is done by adjusting the line linking pointers of the BASIC program while the LOAD is taking place. This pointer readjustment stops when three consecutive zero bytes are encountered, making it possible for machine language programs to be appended to BASIC programs. The relocating LOAD feature will make it possible for us to write and SAVE our BASIC program modules and reload them at a different location.

The non-relocating LOAD is performed by appending a ",1" to the LOAD command after the device number. This tells the operating system to LOAD the program into the same place in memory that it was originally saved from without changing any of the line linking pointers.

THE CLR COMMAND

This command produces the effect of erasing all variables in memory. What it actually does is readjust the pointers listed in Table I. The Start of Arrays and the End of Arrays+1 pointers are reset to the value of the Start of Variables pointer. The Bottom of Strings pointer is reset to the value of the End of Memory pointer. The result is that the BASIC interpreter can no longer find the variables data which was previously stored.

THE NEW COMMAND

This command poroduces the effect of erasing the program from memory. What it actually does is reset the Start of Variables pointer to two bytes above

Continued on page 92

PROGRAM GENERATORS

By Joe Rotello, Jr.

Ever since the introduction of BASIC as a programming language in the mid-1970's, computer users have had to choose between two methods of program procurement: settle for existing "custom" software available off the shelf, or go it alone and attempt to program their own software. This two-sided choice many times became more like a two-bladed knife.

Many times, premade software turned out to be not what the user wanted or needed. Worse yet, much of it did not even work on the computer, and much time and money had to be spent communicating with the software proucer to find a solution (assuming that a solution was there to be found).

Sadly enough, the user who made his own software sometimes faced even greater peril. Most computer users were and still are non-programmers, knowing bits, bytes, random access, CPU, and device number as nebulous quantities. The user often had no idea how to properly plan and program the software he required. Moreover, who had the time?

What is the user to do? He could hire a knowledgeable software programmer. Ahh! Good solution! Until the software is completed and the user finds out that the cost of the programmer's experience has exceeded the cost of the entire hardware system by 700%.

Back to the drawing board. Planning to program his own software, the user attends a programming class at a local place of learning. Oh, oh, our poor user just found out that by the time he picks up sufficient BASIC programming knowledge, America will have space colonies on Mars. Gee, the software had to be ready by next month!

Times *have* changed, haven't they? Now you receive magazines like *Ahoy!* every month and, inside, find hundreds of software programs to choose from. Did we say *hundreds??* We mean that there are *thousands* of Commodore-compatible programs out in softwareland. How many months will it take to sample even ten or fifteen of them? And if you buy one or two that *sound* like the ones you want, will they actually match your needs?

In many instances, the situation reverts to hopelessness and you sadly begin to use your computer less and less until finally it sits on the top shelf in the closet, both you and it a victim of *computopia abundantia,* that dread disease that results from too much hardware and software in too little time.

AND ALONG COMES THE LONE RANGER...

Just when things seem darkest an idea dawns on you: what, just what if *your* computer could be fed an available premade program designed to help you produce your needed program? Let's go on to say that this wonder program would be in the form of easy to read and understand menus, and by answering a series of simple logical questions, the program would eventually understand the kind of program you wish to form. In fact, you can tell this wonder program what data you wish to store, locate, display, or print. It even understands what names or addresses or dates or numbers are important to you.

Ahh, the blood is starting to flow! There is a solution that could, no, *will* expand your understanding and comprehension of the marvelous machine we call the computer! Programs that write other programs for computer users are here, they work well, can be used by everyday users, and are very cost-effective.

ENTER THE PROGRAM GENERATOR

Programs that write other programs are not a new phenomenon in the world of mini and mainframe computers. They do a creditable job and make programmers' work easier and less time-consuming.

When we come down to microcomputers, programs that write other programs have been hard to come by. The main reason we haven't seen them before is that the program code usually took up so

much computer memory that there was little or none left in which to create the desired new program. With the introduction of computers like the C-64, computer memory has been expanded, and with it the likelihood of seeing more really powerful programs being produced for microcomputers.

Programs that write other programs are generally called *program generators* (PG's for short) because that's their purpose: the generation of a running, problem-free program directly related to the wishes and needs of the user or programmer who created it.

Many people still consider PG's hard to work with, probably because of their ability to produce many complex programs that may be large in physical size. A closer look reveals that the microcomputer PG's have evolved into clear, concise, user-friendly programs that put the inherent speed and data manipulation power of the microcomputer to work for the user.

The microcomputer PG has another pleasing side to it. Finally, the power of a mainframe computer program generator is available at a personal computer price. In fact, the retail prices of the C-64-compatible PG's that we reviewed for this article centered around $99.00. Agreed, not a pittance when compared to a computer that retails for under $200, but still an excellent buy considering the ease of use and powerful program creation features these "new generation" PG's possess.

The operation of PG's can be summarized as follows:

1) Programs are indeed created by the user, but the PG dictates the precise form and overall structure of the BASIC code that will eventually be generated.
2) Program creation is generally conducted via user screen menus that extract information about the proposed program's title, content, and basic form directly from the user through a variety of detailed question and answer sessions.
3) The PG eventually outputs a working BASIC program to a disk drive for final storage. The result program is now a "stand alone." The PG is no longer needed in order to load and run the new program.

Although most of the available PG's share a similar procedure structure, the internal methods employed vary. Some packages will be themselves mostly or entirely in machine language. This may or may not enhance the speed and versatility of the product. In fact, being in machine language (the native language of the computer itself) the PG program cannot be listed to the videoscreen or printer, nor can it be modified in any way. This "no list" limitation is generally desirable due to the risk of PG damage that an unwary user might cause through tampering with the program generator itself.

Other PG's will be themselves coded mostly or completely in BASIC, which might make sense to many users. If the program generator's job is to produce a BASIC program, then is it not logical to have the program generator itself be a BASIC program? But those PG's that are in BASIC may tend to be slower in operation than the machine language versions. (The PG's in BASIC that we examined do not exhibit any alarming increase in execution times.)

Another item the user will quickly notice about microcomputer PG's is that in order to gain the enormous programming power required, the PG is not structured as one massive microcomputer program. In fact, the PG is generally made of many discrete sub-programs, spread out over one or more disks, each sub-program charged with producing a discrete but eventually interwoven BASIC program component.

For example, one PG sub-program section might deal exclusively with generating menus or other user information displays for the intended new BASIC program. Another might handle disk or other file structure and data or information input/output code. Still another may delve into the printer needs and the report-generating procedures required.

Hence, one can quickly see that the real art in producing a cohesive, usable BASIC program with a PG lies in the ability of the PG to successfully combine bits and pieces of smaller sections of BASIC code into one program. In many respects, the PG will actually follow procedures similar to what any good programmer would when creating a BASIC program from scratch: initial planning and layout, followed by information gathering, then any disk- or file-related operations required, finally followed by reports or other printer-related coding that may be required by the BASIC program.

As you may have realized by now, the computing power of the PG enables the new or introductory level computer user to produce a fairly complex and functional BASIC program by leading you through the many required steps and processes one by one. It simplifies the procedure, so that assembling your own program actually takes less time than winging it alone.

True, if one wishes to create a haphazard computer program and utilizes a PG to do so, the results will be less than pleasing. However, if you're willing to plan the program you are to create and follow the simple PG procedures, program generating software may well open a whole dimension in microcomputing for you.

USER APPLICATIONS

The applications arena for the Program Generator could quite literally be endless. Operations such as data base implementaion, customer files, mailing lists, inventory management, and accounting programs now lie within the grasp of the average computer user. The range of users is likewise great. Anyone from the passive homeowner computerist to

the serious day-to-day programmer will find the task of program function, layout, and actual implementation made far easier. Instead of spending most of their time debugging their BASIC computer programs, users and programmers alike will be able to devote valuable time to defining just the program that is required to fill their particular need.

PRODUCT OVERVIEW

We'll now inspect a couple of the PG products presently available for Commodore users.

Title: *FileWriter 2* (Formerly *CodeWriter)*
Retail: $99.99
Versions: C-64 (also Apple, IBM PC, Atari, Victor, Kaypro II)
Supplier: CodeWriter Corporation (formerly Dynatech Microsoftware)
7847 N. Caldwell Ave., Niles, IL 60648
(800-621-4109).

FileWriter 2 is the more simple to use PG we have seen. The program seems more suited to the entry-level computer user, although more advanced programmers will find it powerful enough for many applications. The program manual is excellently written, and easy for even the novice to comprehend. Its only drawback is that only one detailed example of PG use is given (although this single example is very well detailed). A more definitive user manual should be available soon. *FileWriter 2* is itself programmed mostly in BASIC, although there seem to be internal references to a few computer-specific operating system routines. The PG overall execution speed is acceptable for most user applications.

In our test usage, we found *FileWriter 2* pleasant to use. The program does not presume that the user understands anything more complicated than how to answer well-put specific questions presented to the operator via onscreen menus.

Although it is impossible to fairly time the programming efficiency that *FileWriter 2* gives, we noted that to duplicate an already existing, fairly complex hand coded program took about 45 minutes versus the estimated 15 man hours the hand coded version required.

Title: *The Last One*
Retail: $99.95
Versions: C-64 (also PET/CBM, Apple, IBM PC, TRS-80 Mod II, Victor, Kaypro II)
Supplier: Computer Marketing Services
300 W. Marlton Pike, Cherry Hill, NJ 08002 (800-222-0585).

The Last One has been available in Great Britian for over two years, initially on the Commodore PET/CBM computers. It is the more thorough and internally complex of the two program generators



***The Last One** is itself programmed mostly in machine language, although there seem to be a few internal program references to BASIC where the use of BASIC is preferred. READER SERVICE NO. 281*

mentioned here, in respect to the more complete treatment of programming introduction, data files, sorting, and report generation both in the user manual and in the actual program. The present user manual is somewhat hard to read due to its small size and printing, but most first-time users will not find it hard to comprehend, even where more complex programming topics are discussed.

Again, as with *FileWriter 2,* a new, more spacious user manual is being prepared and it also should be available by the time you read this article.

The Last One appears well-suited to most areas of programming, although many first-time users will be somewhat alarmed at the thickness of the present user manual. The manual's size merely makes the program seem more complex than it actually is. More advanced programmers will find both manual and program powerful enough for just about any programming need.

The Last One is itself programmed mostly in machine language, although there seem to be a few internal program references to BASIC where the use of BASIC is preferred. Overall program execution speed is acceptable for any user application.

In our test usage, we found *The Last One* pleasant to use, and the program does not presume that the user understands anything more complicated than how to answer well-put specific questions presented to the operator via onscreen menus. If this last statement sounds familiar, it should. Both of the available program generators are *very* menu-oriented, making the overall program operation far easier for any user.

Again, it is impossible to fairly time the programming efficiency that *The Last One* gives. However, we noted that to duplicate an already-existing hand coded program timed in at about 30 minutes, versus the estimated 10 hours the hand coded version required.

MAKING YOUR OWN PROGRAM

Let's take a trip through a representative program generator and see how the process of program generation is accomplished. *The Last One* (*TLO* for short) will serve as our vehicle.

The first step is to read the entire user manual carefully. Do not attempt to absorb everything in it at once. Even experienced PG users will never have to use all the commands made available in a program like *TLO,* so don't go overboard.

We carefully followed all manual directions and created backup copies of all the important program and file disks. This did not take long, as the entire process is under the control of the *TLO* program.

From that point on, we followed the example introductory program found in the tutorial section of the *TLO* manual. We found that all parts of the *TLO* are accessed from a Main Dispersal Menu (MDM):

```
Create program ......... <1>
Modify program ......... <2>
Modify file............. <3>
Define file............. <4>
Enquiry................. <5>
Certify workdisk........ <6>
Resume.................. <7>
Printer specification... <8>
Return to BASIC......... <9>
```

The next step was to "Certify Workdisk" or, as it is better known to Commodore users, HEADER and format a disk that would hold the workfiles for the program and files that we are going to create. Again, the user follows screen instructions and the disk is created and "certified" by the *TLO* system. Next, we advised *TLO* what printer, if any, will be used with the PG system. The available choices are 1) No printer, 2) Standard Commodore printer, or 3) A standard ASCII printer. This information is then stored on the *TLO* disk for future reference.

Our next move was to lay out on paper, in plain English, a short list of the major points we will have in the program (sometimes this can be done in your head). In our case, we followed the *TLO* information in the manual. Our goal was to create a simple telephone book file system. Among the major points were a nice menu, a routine to add records into the file, the ability to review and correct the records *before* writing them to the file, the ability to search the file for a simple match, and the provision to print out an uncomplicated hard copy report that would list name, city, and telephone of each record in the file.

This "laying out" of the program to be generated is very important. In fact, a slightly modified version of this initial program layout will be used by *TLO* to actually build our user program and data files for us a little bit later into the *TLO* process.

Next, we advised the *TLO* that our intended program would use data files. Understanding this, *TLO* proceeds to lead us, in a very user-friendly manner, through the routine that allows us to set up our file data fields. The file fields we build into the program

Continued on page 92

CREATING YOUR OWN GAMES

From page 18

all the arithmetic *before* we erase the old position, so that the new position is PRINTed *immediately* after the old position is blanked.

Actually, that program line wouldn't work, because the new character would be PRINTed on the next line. We haven't taken care of vertical movement.

VERTICAL POSITION

To PRINT the figure in the right row, we just use the right number of cursor-down characters—CHR$(17). Of course, to end up on the right line, we have to count down from the top of the screen each time. That means that before we PRINT the right number of cursor-downs, we have to begin by PRINTing a HOME character—CHR$(19).

The drawback with using PRINT for movement is that because of the way the computer formats the screen, you can't PRINT in the rightmost column or the bottom row without causing complications to the screen display. It's a simple matter, however, to avoid using those rows, and the player never knows the difference if you make the border and background the same color.

Screen size is one of the few places where the VIC and 64 differ in this program. The 64 can have 25 rows, and the VIC can only have 23. And the 64 allows 40-character lines, while the VIC only allows 22-character lines. To handle this in the program, we'll set the variable BE to the value of the bottom edge of the screen and RE to the value of the right edge of the screen. We number the possible row numbers on the VIC from 0 to 22. Since we can't use the last row, BE will be set to 21. With the 64, BE will be set to 23. On the VIC, the column numbers are 0 to 21; since we can't use the last position, RE is set to 20. With the 64, RE will be set to 38.

Once BE has been set, we can set up an array, VM$(*n*), to hold the vertical movement strings, using this subroutine:

```
500 DIM VM$(BE):VM$(0)=CHR$(19)
510 FOR I=1 TO BE:VM$(I)=VM$(I-1)
+CHR$(17):NEXT I
540 RETURN
```

Line 500 DIMensions the string to the number of lines on the screen (BE). Then the 0 position is set to the HOME character, CHR$(19). In line 510, the rest of the variables are set. Each one will begin with the HOME character, and each will have one more cursor-down character—CHR$(17)—than the one before. From now on, when we PRINT VM$(5), the cursor will end up on row 5; when we PRINT VM$(19), the cursor will end up on row 19.

Now, in combination, we have complete control over the horizontal and vertical position of the figure. The new vertical position is held in the variable V and the old vertical position in the variable VX, which are used in the array variable, like this:

```
PRINT VM$(VX)TAB(HX)" "VM$(V)TAB(H
)PF$
```

Notice that a single PRINT statement moves the cursor to the old row (VX) and column (HX), PRINTs a blank, then moves the cursor to the new row(V) and the new column (H) and PRINTs the figure, PF$.

GETTING THE MOVEMENT UNDER CONTROL

The movement loop is the heart of an action game. It begins by getting instructions from the player. If the player wants to move, you test to see if the move is legal; if it is, you carry it out. Then you go back and get more instructions.

For our game, we'll use the f5 and f7 keys to control vertical movement—f5 is up and f7 is down. We'll use the SHIFT and Commodore logo keys to control horizontal movement—Commodore is left and SHIFT is right.

There are good reasons for using these keys. Since SHIFT and Commodore are read at location 653 and f5 and f7 are read at location 197, the player can signal a horizontal and a vertical movement *at the same time.* This is crucial, because it allows diagonal movement. In a maze game, where the only choices are up, down, left, and right, and diagonals are impossible, you wouldn't need to include diagonals, and then any four keys would do. But we want our figure to move freely around the screen.

Remember, to move left, we want to subtract 1 from H; to move right, we add 1.

If SHIFT is pressed, there will be a 1 at location

Next month in *Ahoy!,* Orson Scott Card examines *The World of the Game*—generating a screen display that will make players want to stay in the world that you create!

653; if Commodore is pressed there will be a 2. Those are the two values that matter. Any other value should result in no movement. We could handle this with program lines like this:

```
LR=PEEK(653)
IF LR=1 THEN H=H+1
IF LR=2 THEN H=H-1
```

This would work, but it would be slow. It's much easier to set up an array, HM(*x*), in which HM(1)=1 and HM(2)=-1, and all the other elements equal 0. Likewise, we could set up an array VM(*x*) to handle vertical movement. Since location 197 can return values from 0 to 64, there'll be 64 elements in the array. With the 64, f5 returns a value of 6, which means *up;* f7 returns a value of 3, which means down. So the array element VM(6) is set to −1 and VM(3) is set to 1—all the other values of VM(*x*) are set to 0.

Now we can read the values and set the new values of H and V with a minimum of arithmetic:

```
100 V=VX+VM(PEEK(197)):H=HX+HM(PE
EK(653))
```

There it is—V and H have been set directly from the values found in locations 197 and 653, using the values found there as indexes into preset arrays.

It's time to put all this together into a program. There are some things in this program that I haven't explained yet—I'll explain what they're for afterward. This program is for the Commodore 64; the values to turn it into a VIC program are in REM statements along the way. The differences are slight.

```
5 DIM HM(7),VM(64)
10 GOSUB 600:GOSUB 500
80 V=10:H=10
90 GOTO 140
97 REM
98 REM MAIN MOVEMENT LOOP
99 REM
100 V=VX+VM(PEEK(197)):H=HX+HM(PE
EK(653))
110 IF H=HX AND V=VX THEN 100
120 IF V>BE THEN V=V-BF
121 IF V<0 THEN V=V+BF
122 IF H>RE THEN H=H-RF
123 IF H<0 THEN H=H+RF
140 PRINT VM$(VX)TAB(HX)" "VM$(V)
TAB(H)PF$
150 HX=H:VX=V
190 GOTO 100
497 REM
498 REM SET UP MOVEMENT STRINGS
499 REM
500 DIM VM$(BE):VM$(0)=CHR$(19)
510 FOR I=1 TO BE:VM$(I)=VM$(I-1)
+CHR$(17):NEXT
540 RETURN
597 REM
598 REM SETUP FOR COMMODORE 64
599 REM
600 RE=38:REM VIC VALUE IS 20
605 RF=39:REM VIC VALUE IS 21
610 BE=23:REM VIC VALUE IS 21
615 BF=24:REM VIC VALUE IS 22
620 PF$=CHR$(122)
630 POKE 53281,0:POKE 53280,0
631 REM VIC LINE: 630 POKE 36879,
8
640 POKE 657,128:PRINT CHR$(147)
650 FOR I=0 TO 64:VM(I)=0:NEXT:FO
R I=0 TO 7:HM(I)=0:NEXT
660 HM(1)=1:HM(2)=-1:VM(3)=1:VM(6
)=-1
661 REM VIC LINE: 660 HM(1)=1:HM(
2)=-1:VM(63)=1:VM(55)=-1
690 RETURN
```

Line 5 DIMensions the arrays HM(*n*) and VM(*n*) to hold the keyboard data. Line 10 sends the program to the two setup routines at 600 and 500. Line 80 sets the starting values for the vertical and horizontal position variables. Line 90 routes the program into the middle of the movement loop, so the figure will appear on the screen when the program starts.

Lines 100 through 190 are the movement loop. In line 110, if the player hasn't called for any movement, the program is routed back to line 100.

Lines 120 through 123 check to see if the movement has taken the figure beyond the edge of the screen. If, for example, the figure is off the bottom of the screen—V>BE—then the value BF, which is one greater than BE, is subtracted from V. This puts the figure at the top of the screen. If V is less than 0, BF is added to V, which puts the figure at the bottom of the screen.

These four lines can be replaced by a single line:

```
120 V=V+BF*((V>BE)-(V<0)):H=H+RF*
((H>RE)-(H<0))
```

This one line, however, actually takes up more time than lines 120, 121, 122, and 123, because six ar-

ithmetic operations must be performed each time through the loop, whether the figure is beyond an edge or not. It slows down the program only slightly, but the difference is noticeable.

Lines 120 through 123 make the figure "wrap around" the screen—that is, when the figure moves off the bottom of the screen, it reappears at the top; when it moves offscreen to the left, it reappears at the right. If you want, you can change the program so the figure simply stops at the edge. If you want to stop at the edges, use these four lines:

```
120 IF V>BE THEN V=V-1
121 IF V<0 THEN V=V+1
122 IF H>RE THEN H=H-1
123 IF H<0 THEN H=H+1
```

Instead of moving the figure to the opposite edge of the screen, these lines make the figure back up one position if it has moved beyond the edge.

Line 140 actually erases the figure at the old position (HX and VX) and PRINTs it at the new position (H and V).

Line 150 sets HX to the new value of H and VX to the new value of V. Then, when the player calls for the next movement, HX and VX will be used to erase the figure at its current position.

Line 190 completes the loop, sending the program back to line 100 to get the player's next instruction.

The rest of the program is setup. Lines 500 to 540 set up the vertical movement string array VM$(*x*). Lines 600 to 690 set up everything else.

Lines 600 to 615 set BE, BF, RE, and RF to the values needed to keep the figure from moving beyond the edge of the screen. If you have a VIC, change the numbers to the values in the REM statements.

Line 620 sets PF$ to CHR$(122), the diamond character. You can change this to any value that you want, of course.

Line 630 changes the screen and border to black. If you have a VIC, replace line 630 with the line after the REM statement in line 631.

Line 640 disables the SHIFT-Commodore feature and clears the screen. (If you don't disable the SHIFT-Commodore feature, then when you move left and right you'll change from upper to lower case and back again—not a helpful addition to the game!)

Lines 650 and 660 set the values of the keyboard reading arrays, VM(*x*) and HM(*x*). If you have a VIC, replace line 660 with the line after the REM statement in line 661. (And if you have a 64, of course you don't need to include lines 631 and 661.) Line 690 ends the subroutine.

COMPLEX CHARACTERS

Sometimes you'll want to move a figure that's larger than a single character. This program can easily be adapted to let you move complex characters. Let's create a complex character that is three characters high and three characters wide. First, since the character is wider and taller than before, the variables BE, BF, RE, and RF must be set to smaller limits. The original figure was one character wide and high; the new one is three characters wide and three characters high, the difference in each direction is 2. So lines 600 through 615 are now:

```
600 RE=36:REM VIC VALUE IS 18
605 RF=37:REM VIC VALUE IS 19
610 BE=21:REM VIC VALUE IS 19
615 BF=22:REM VIC VALUE IS 20
```

To create the complex figure PF$ and the blank figure to erase it, BL$, these four lines replace line 620:

```
620 AA$=CHR$(17)+CHR$(157)+CHR$(1
57)+CHR$(157)
623 PF$=CHR$(110)+CHR$(118)+CHR$(
109)+AA$+CHR$(110)+CHR$(113)+CHR$
(109)+AA$
625 PF$=PF$+CHR$(109)+CHR$(164)+C
HR$(110)
627 BL$="   "+AA$+"   "+AA$+"   "
```

And, finally, change line 140 to:

```
140 PRINT VM$(VX)TAB(HX)BL$VM$(V)
TAB(H)PF$
```

Just don't ask me what the figure is supposed to represent—as nearly as I can tell, it's a one-eyed bunny rabbit.

THIS IS NOT A GAME

All we've got so far is some shapes moving around on the screen. But the shapes don't mean anything, and moving around doesn't accomplish anything. A good game tells a story, has visual appeal, provides rewards for good play, and has tasks to carry out, obstacles to overcome, and puzzles to solve.

Next month we'll create background objects on the screen—and put some of them in motion. In the meantime, if there's anything you'd like to say—about games, this column, or anything at all—feel free to write to me in care of *Ahoy!* It'll be easier to write about things that interest you if you tell me what you want to see. □

COMMODORE 64
(more power than Apple II at half the price)

$99.50*

- 170K DISK DRIVE $159.00 *
- TRACTION FRICTION PRINTER $99.00 *

COMPUTER AND SOFTWARE SALE

WE HAVE THE BEST SERVICE

WE HAVE THE LOWEST PRICES

VIC-20
(a real computer at the price of a toy)

$79.50*

- 40-80 COLUMN BOARD $59.00
- 32K RAM EXPANDER $79.00

* COMMODORE 64 COMPUTER $99.50
You pay only $199.50 when you order the powerful 84K COMMODORE 64 COMPUTER! LESS the value of the SPECIAL SOFTWARE COUPON we pack with your computer that allows you to SAVE OVER $100 off software sale prices!! With only $100 of savings applied, your net computer cost is $99.50!!

SOFTWARE BONUS PACK $24.95
When you buy the Commodore 64 Computer from Protecto Enterprizes you qualify to purchase ONE SOFTWARE BONUS PACK for a special price of $24.95!! Normal price is $49.95 (40 programs on disk or 24 programs on 5 tapes).

*170K DISK DRIVE $159.00
You pay only $259.00 when you order the 170K Disk Drive! LESS the value of the SPECIAL SOFTWARE COUPON we pack with your disk drive that allows you to SAVE OVER $100 off software sale prices!! With only $100 of savings applied, your net disk drive cost is $159.00.

* TRACTION FRICTION PRINTER $99.00
You pay only $199.00 when you order the Comstar T/F deluxe line printer that prints 8 1/2 x 11 full size, single sheet, roll or fan fold paper, labels etc. 40, 66, 80, 132 columns. Impact dot matrix, bi-directional, 80 CPS. LESS the value of the SPECIAL SOFTWARE COUPON we pack with your printer that allows you to SAVE OVER $100 off software sale prices!! With only $100 of savings applied your net printer cost is only $99.00.

80 COLUMN BOARD $99.00
Now you program 80 COLUMNS on the screen at one time! Converts your Commodore 64 to 80 COLUMNS when you plug in the 80 COLUMN EXPANSION BOARD!! List $199—PLUS—you also can get an 80 COLUMN BOARD WORD PROCESSOR with mail merge, terminal emulator, ELECTRONIC SPREAD SHEET. List $59.00 **SALE $24.95** if purchased with 80 COLUMN BOARD!! (Tape or Disk)

80 COLUMNS IN COLOR EXECUTIVE WORD PROCESSOR $69.00
This EXECUTIVE WORD PROCESSOR is the finest available for the COMMODORE 64 computer! The ULTIMATE for PROFESSIONAL Word-processing application! DISPLAYS 40 OR 80 COLUMNS IN COLOR or Black and White! Simple to operate, powerful text editing with a 250 WORD DICTIONARY, complete cursor and insert/delete key controls line and paragraph insertion, automatic deletion, centering, margin settings and output to all printers! Includes a powerful mail merge. List $99.00 20,000 WORD DICTIONARY - List $24.95 **SALE $19.95.** EXECUTIVE DATA BASE - List $69.00 **SALE $49.00.** (Disk only).

SPECIAL SOFTWARE COUPON

We pack a SPECIAL SOFTWARE COUPON with every COMMODORE 64 COMPUTER-DISK DRIVE-PRINTER-MONITOR we sell! This coupon allows you to SAVE OVER $100 OFF SALE PRICES! Up to $500 savings are possible!!

PROFESSIONAL SOFTWARE COMMODORE 64

Name	List	Sale	Coupon
Executive Word Processor	$99 00	$69 00	$59 00
Executive Data Base	$69 00	$59 00	$39 00
20,000 Word Dictionary	$24 95	$19 95	$14 95
Electronic Spreadsheet	$59 95	$49 00	$39 00
Accounting Pack	$49 00	$39 00	$29 00
Total 5.2 Word Processor			
Tape	$69 00	$49 00	$34 00
Disk	$79 95	$59 00	$39 00
Total Text 2.6 Word Processor			
Tape	$44 95	$34 95	$22 00
Disk	$49 00	$39 00	$27 00
Total Label 2.6			
Tape	$24 95	$18 00	$12 00
Disk	$29 95	$23 00	$15 00
Programmers Helper (Disk)	$59 00	$39 95	$29 95
80 Column Screen (Disk)	$59 95	$39 95	$29 95
Crush-Crumble-Chomp (Tape/Disk)	$29 95	$24 95	$19 95
Pitstop (Cartridge)	$39 95	$29 95	$24 95
Typing Teacher (Tape/Disk)	$29 95	$24 95	$15 00
Sprite Designer (Disk)	$16 95	$14 95	$10 00
Fireball Joy Stick	$24 95	$15 95	$10 00
Light Pen	$39 95	$16 95	$14 95
Dust Cover	$ 8 95	$ 6 95	$ 4 60

(See 100 coupon items in our catalog!)

Write or call for
Sample SPECIAL SOFTWARE COUPON!

EXECUTIVE QUALITY PROFESSIONAL BUSINESS SOFTWARE

The Cadillac of business programs for Commodore 64 Computers

Item	List	*SALE	Coupon
Inventory Management	$99 00	$59 00	$49 00
Accounts Receivable	$99 00	$59 00	$49 00
Accounts Payable	$99 00	$59 00	$49 00
Payroll	$99 00	$59 00	$49 00
General Ledger	$99 00	$59 00	$49 00

VIC-20 COMPUTER $79.50
This 25K VIC-20 computer includes a full size 66 key typewriter keyboard color and graphics keys, upper/lower case, full screen editor, 16K level II microsoft basic, sound and music, real time floating point decimal, self teaching book, connects to any T.V. or monitor! (Limit one to a customer!)

40-80 COLUMN BOARD $59.00
Now you can get 40 OR 80 COLUMNS on your T.V. or monitor at one time! No more running out of line space for programming and making columns! Just plug in this Expansion Board and you immediately convert your VIC-20 computer to 40 OR 80 COLUMNS!! You can also get an 80 COLUMN BOARD WORD PROCESSOR with mail merge, terminal emulator, ELECTRONIC SPREAD SHEET!! List $59.00. **SALE $24.95** if purchased with 80 COLUMN BOARD! (Tape or Disk).

32K RAM EXPANDER $79.00
This cartridge increases programming power over **8 times!!** Expands total memory to 57K (57,000 bytes) almost Com-64 power! Block switches are on outside of cover! Has expansion part!! Lists for $199 **(OUR BEST BUY!)**

60K MEMORY EXPANDER $49.00
Sixslot Board — Switch selectable — Reset button — Ribbon cable — CARDCO. A must to get the most out of your VIC-20 Computer!

8K RAM CARTRIDGE $39.00
Increases programming power 2 1/2 times. Expands total memory to 33K (33,000 bytes). Memory block switches are on outside of cover! Includes FREE $16.95 game.

16K RAM CARTRIDGE $49.00
Increases programming power 4 times. Expands total memory to 41K (41,000 bytes). Memory block switches are an outside cover! CARDCO Includes FREE $29.95 adventure game!!

9" GREEN SCREEN MONITOR $69.00
Excellent quality SANYO, easy to read, 80 columns x 24 lines, Green Phosphorous screen with anti-glare, metal cabinet! Saves your T.V. PLUS $9.95 for connecting cable.

12" GREEN OR AMBER MONITOR $99.00
Your choice of green or amber screen monitor, top quality, SANYO, 80 columns x 24 lines, easy to ready, anti-glare, faster scanning! A must for word processing PLUS $9.95 for connecting cable.

• LOWEST PRICES • 15 DAY FREE TRIAL • 90 DAY FREE REPLACEMENT WARRANTY
• BEST SERVICE IN U.S.A. • ONE DAY EXPRESS MAIL • OVER 500 PROGRAMS • FREE CATALOGS

Add $10.00 for shipping, handling and insurance. Illinois residents please add 6% tax. Add $20.00 for CANADA, PUERTO RICO, HAWAII orders. WE DO NOT EXPORT TO OTHER COUNTRIES.

Enclose Cashiers Check, Money Order or Personal Check. Allow 14 days for delivery, 2 to 7 days for phone orders, 1 day express mail! Canada orders must be in U.S. dollars. VISA — MASTER CARD — C.O.D.

PROTECTO
ENTERPRIZES (WE LOVE OUR CUSTOMERS)
BOX 550, BARRINGTON, ILLINOIS 60010
Phone 312/382-5244 to order

Reader Service No. 256

SANYO MONITOR SALE!!



9″ Data Monitor

- 80 Columns × 24 lines
- Green text display
- Easy to read - no eye strain
- Up front brightness control
- High resolution graphics
- Quick start - no preheating
- Regulated power supply
- Attractive metal cabinet
- UL and FCC approved

- ***15 Day Free Trial - 90 Day Immediate Replacement Warranty***

9″ Screen - Green Text Display	**$ 69.00**
12″ Screen - Green Text Display (anti-reflective screen)	**$ 99.00**
12″ Screen - Amber Text Display (anti-reflective screen)	**$ 99.00**
12″ Screen-Super 1000 Line Amber Text Display	**$129.00**
14″ Screen - Color Monitor (national brand)	**$249.00**

Display Monitors From Sanyo

With the need for computing power growing every day, Sanyo has stepped in to meet the demand with a whole new line of low cost, high quality data monitors. Designed for commercial and personal computer use. All models come with an array of features, including up-front brightness and contrast controls. The capacity 5 × 7 dot characters as the input is 24 lines of characters with up to 80 characters per line.

Equally important, all are built with Sanyo's commitment to technological excellence. In the world of Audio/Video, Sanyo is synonymous with reliability and performance. And Sanyo quality is reflected in our reputation. Unlike some suppliers, Sanyo designs, manufactures and tests virtually all the parts that go into our products, from cameras to stereos. That's an assurance not everybody can give you!

SANYO

Official Video Products of the Los Angeles 1984 Olympics

• LOWEST PRICES • 15 DAY FREE TRIAL • 90 DAY FREE REPLACEMENT WARRANTY
• BEST SERVICE IN U.S.A. • ONE DAY EXPRESS MAIL • OVER 500 PROGRAMS • FREE CATALOGS

Add $10.00 for shipping, handling and insurance. Illinois residents please add 6% tax. Add $20.00 for CANADA, PUERTO RICO, HAWAII orders. WE DO NOT EXPORT TO OTHER COUNTRIES.

Enclose Cashiers Check, Money Order or Personal Check. Allow 14 days for delivery, 2 to 7 days for phone orders, 1 day express mail! Canada orders must be in U.S. dollars. Visa - MasterCard - C.O.D.

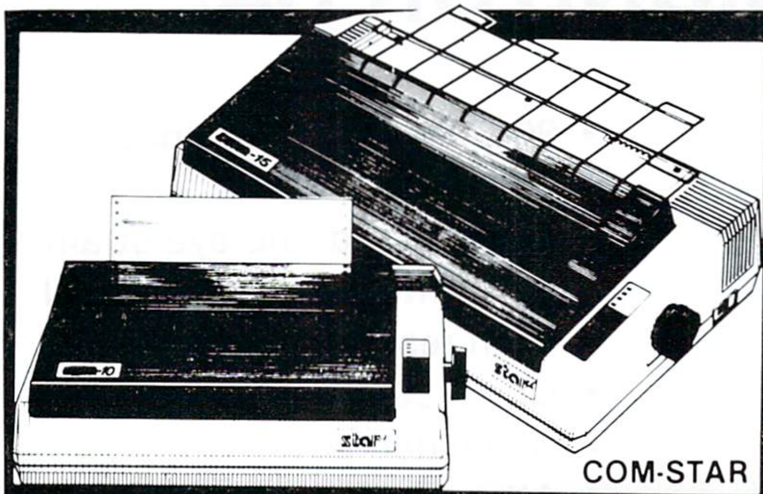
PROTECTO

ENTERPRIZES (WE LOVE OUR CUSTOMERS)

BOX 550, BARRINGTON, ILLINOIS 60010
Phone 312/382-5244 to order

Reader Service No. 257

80 COLUMN PRINTER SALE—$149.00*



COM-STAR T/F

Tractor Friction Printer

only $199**

•15 Day Free Trial -180 Day Immediate Replacement Warranty

- Lowest Priced, Best Quality, Tractor-Friction Printers in the U.S.A.
- Fast 80-120-160 Characters Per Second
- 40, 46, 66, 80, 96, 132 Characters Per Line Spacing
- Word Processing
- Print Labels, Letters, Graphs and Tables
- List Your Programs
- Print Out Data from Modem Services
- "The Most Important Accessory for Your Computer"

*STX-80 COLUMN PRINTER—$149.00

Prints full 80 columns. Super silent operation, 60 CPS, prints Hi-resolution graphics and block graphics, expanded character set, exceptionally clear characters, fantastic print quality, uses inexpensive thermal paper! Best thermal printer in the U.S.A.! (Centronics Parallel Interface).

**DELUXE COMSTAR T/F 80 CPS PRINTER—$199.00

The COMSTAR T/F (Tractor Friction) PRINTER is exceptionally versatile. It prints 8½" x 11" standard size single sheet stationary or continuous feed computer paper. Bi-directional, impact dot matrix, 80 CPS, 224 characters. (Centronics Parallel Interface).

Premium Quality—120 CPS COMSTAR T/F SUPER-10X PRINTER—$289.00

COMSTAR T/F (Tractor Friction) SUPER-10X PRINTER gives you all the features of the COMSTAR T/F PRINTER plus a 10" carriage, 120 CPS, 9 x 9 dot matrix with double strike capability for 18 x 18 dot matrix (near letter quality), high resolution bit image (120 x 144 dot matrix), underlining, back spacing, left and right margin settings, true lower decenders with super and subscripts, prints standard, italic, block graphics and special characters, plus 2K of user definable characters! The COMSTAR T/F SUPER-10X PRINTER was Rated No. 1 by "Popular Science Magazine." It gives you print quality and features found on printers costing twice as much!! (Centronics Parallel Interface) (Better than Epson FX 80).

Premium Quality—120 CPS COMSTAR T/F SUPER-15½" PRINTER—$379.00

COMSTAR T/F SUPER 15½" PRINTER has all the features of the COMSTAR T/F SUPER-10X PRINTER plus a 15½" carriage and more powerful electronics components to handle large ledger business forms! (Better than Epson FX 100).

Superior Quality SUPER HIGH SPEED—160 CPS COMSTAR T/F 10" PRINTER—$489.00

SUPER HIGH SPEED COMSTAR T/F (Tractor Friction) PRINTER has all the features of the COMSTAR SUPER-10X PRINTER plus SUPER HIGH SPEED PRINTING—160 CPS, 100% duty cycle, 8K buffer, diverse character fonts, special symbols and true decenders, vertical and horizontal tabs. RED HOT BUSINESS PRINTER at an unbelievable low price!! (Serial or Centronics Parallel Interface)

Superior Quality SUPER HIGH SPEED—160 CPS COMSTAR T/F 15½" PRINTER—$579.00

SUPER HIGH SPEED COMSTAR T/F 15½" PRINTER has all the features of the SUPER HIGH SPEED COMSTAR T/F 10" PRINTER plus a 15½" carriage and more powerful electronics to handle larger ledger business forms! Exclusive bottom paper feed!!

PARALLEL INTERFACES

For VIC-20 and COM-64—$49.00
For All Apple Computers—$79.00

NOTE: Other printer interfaces are available at computer stores!

Double Immediate Replacement Warranty

We have doubled the normal 90 day warranty to 180 days. Therefore if your printer fails within "180 days" from the date of purchase you simply send your printer to us via United Parcel Service, prepaid. We will IMMEDIATELY send you a replacement printer at no charge, prepaid. This warranty, once again, proves that WE LOVE OUR CUSTOMERS!

Add $17.50 for shipping, handling and insurance. WE DO NOT EXPORT TO OTHER COUNTRIES EXCEPT CANADA.

Enclose Cashiers Check, Money Order or Personal Check. Allow 14 days for delivery, 2 to 7 days for phone orders, 1 day express mail! Canada orders must be in U.S. dollars. VISA — MASTER CARD ACCEPTED. We ship C.O.D.

PROTECTO ENTERPRIZES (WE LOVE OUR CUSTOMERS)
BOX 550, BARRINGTON, ILLINOIS 60010
Phone 312/382-5244 to order

SUPER-10" ABCDEFGHIJKLMNOPQRSTUVWXYZ
ABCDEFGHIJKLMNOPQRSTUVWXYZ 1234567890

BOOK REVIEW

COMMODORE 64 PROGRAMS FOR THE HOME By Charles D. Sternberg

Commodore 64 Programs for the Home (Hayden Book Company, 1983) is a model of what book collections of programs should be. It includes 39 programs which cover a wide variety of applications. The listings are clear and include adequate explanations for even the beginning user.

The book starts with a list of required equipment. The author assumes that most Commodore users have tape drives, but refers readers to an appendix for help if they wish to use disk drives. The introduction explains how each section is organized.

Sternberg summarizes the basics of entering, debugging, and using the programs. He notes such pitfalls as mixing up "O" and zero, and tells how to make corrections. This section would be improved by more explicit instructions on entering DATA statements. Many of the programs require that the user substitute his own data for sample data given in the program listings.

The author provides an appendix with notes on Commodore BASIC so people with other kinds of computers can adapt the programs. He does not mention the VIC 20, but most of the programs do not use graphics or sound, so they should work on the VIC.

The programs themselves are varied. About a third of the programs record information by adding DATA statements to the program listing and saving the new listing after each entry session. This simple mechanism is used for recording family history, remembering birthdays, planning meals and family chores, and keeping records on the family car.

Mom and Dad can compute mortgage payments and interest on savings. The home worker has programs to help record time, make orders, and create bids.

Education programs include an outline generator, flash cards, and a question and answer file for test review. The whole family can keep up a win/loss record for their favorite sports team. A program for keeping track of club dues is a good way to show off the computer to friends.

Each program includes a description, instructions for using it, notes on pitfalls and how to change the program, a program listing, a sample run, and a list of major variables. The listings are full-size, printed on a letter-quality printer. They are easy to read, and the author has sacrificed "crunching" to make them more understandable. (He does tell the user how to crunch.)

The explanations of how the programs work are not detailed enough to help a real novice. However, they are useful to anyone with a little programming experience. The sample runs provide a good example for studying how to lay out the printed page.

This is a solid book. The programs are useful and can readily be adapted for other applications. The format makes the book accessible to the beginner with minimal frustration, yet the programs are meaty enough to interest more advanced users.

—Annette Hinshaw

POST TIME



FOR THE C-64 AND VIC 20*
*4K Expansion Required

By Bob Lloret

(VIC version by Robert Alonso)

Wouldn't it be great if someone gave you $500.00 and told you to go to the races and try your luck? *Post Time* will do just that. This is an arcade horse racing game that brings all the thrill and excitement of real racing into your home. One to four players (only one player in VIC version) will enjoy choosing a horse, making a bet, then cheering their choice all the way to the finish line. Type in the program, then save a copy to disk or tape in case of any typing mistakes.

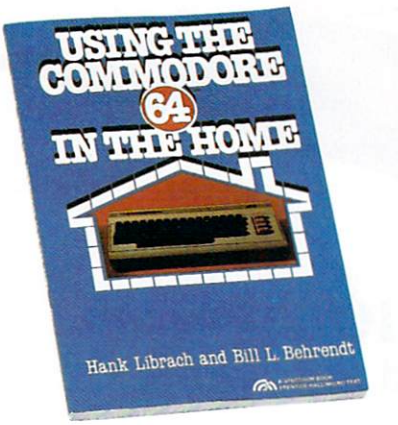
Once your program is entered, type "RUN", then press return. The first display you see is the title and author screen. This screen incorporates a unique routine to display the author's name. Readers interested in learning programming can incorporate this routine in their own programs to enhance displays. After a brief pause, the next screen that appears is for user information. 64 users, type in the number of players (1 to 4), then type in the first name of each player. From here on, the program will prompt all users in turn. The next screen to appear is the instructions. One important part of the instructions to be kept in mind is if you lose all your money (Go Broke), you will not be able to bet any more. After reading the instructions press "fl" and your first race choices will appear with the odds of each choice. At the bottom of the display you will be asked the type of bet you want to make. Press 1 for a Win bet, 2 for a Place bet, 3 for a Show bet and 4 for Trifecta. (VIC players can bet to win only.)

A win bet pays the odds shown, a place bet pays ½ the odds shown, and a show bet pays ⅓ the odds shown. Payoff rules are the same as at the real race track. If you bet place and your horse wins, you win. If you bet show and your horse comes in 1st, 2nd, or 3rd, you win.

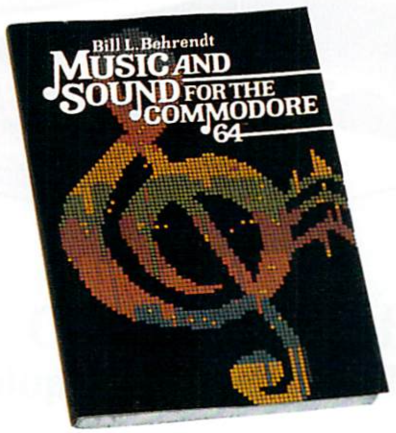
When all players have entered their bets, the screen changes to the race track where the race will take place. All horses will be at the left or starting gate and the race call will sound. After the race

Continued on page 96

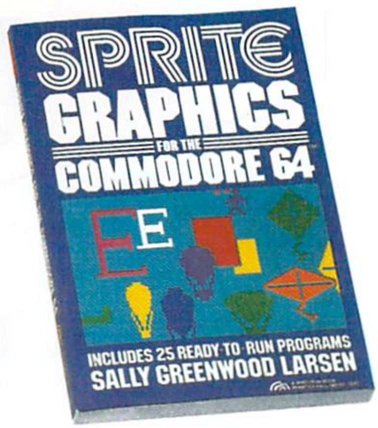
Prentice-Hall speaks a Commodore language other publishers have forgotten. English.*



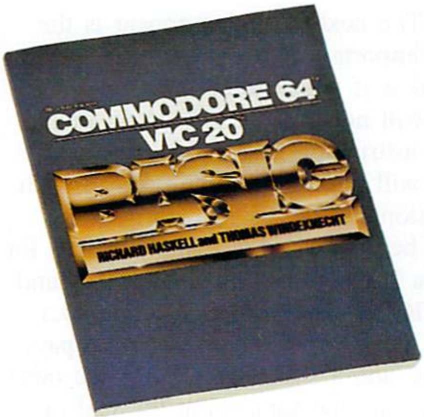
USING THE COMMODORE 64 IN THE HOME by Hank Librach and William Behrendt. Home of the future! Twenty original programs for check book keeping, loan payments, family nutrition, education, games, and much more. Book/disk available. $10.95



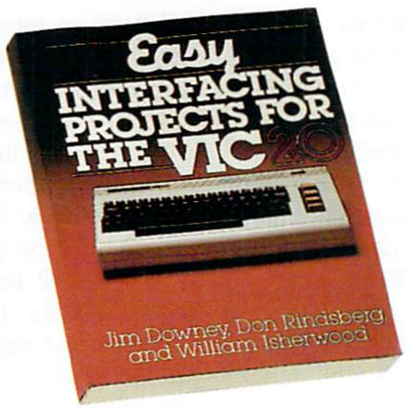
MUSIC AND SOUND FOR THE COMMODORE 64 by Bill L. Behrendt. How to use the Commodore 64's Sound Interface Device and how to write programs that match the sounds of various band instruments. $14.95



SPRITE GRAPHICS FOR THE COMMODORE 64 by Sally Greenwood Larsen. Shows how to produce high resolution, color, animated graphics. Twenty-five ready-to-run sample programs. $15.95



COMMODORE 64/VIC 20 BASIC by Richard Haskell and Thomas Windeknecht. A top-down programming guide, complete with examples illustrated by video screen photos, that introduces the beginner and advanced user alike to the concepts—and actual writing—of programs in BASIC. $13.95



EASY INTERFACING PROJECTS FOR THE VIC-20 by James Downey, Don Rindsberg, and William Isherwood. Dozens of interfacing projects written in BASIC and specifically designed to maximize the VIC-20's power. $12.95



INTRODUCING COMMODORE 64 MACHINE CODE by Ian Sinclair. This clear, step-by-step intro to programming in machine language also includes sophisticated applications in fast moving graphics and games. $12.95

PRENTICE-HALL BOOKWARE™/THE LEADER IN COMPUTER PUBLISHING

For more information about our computer books and software, write to us at the address below. Dealer inquiries welcome.
Prentice-Hall, General Publishing Division, Englewood Cliffs, N.J. 07632

* We guarantee that all our guides are easy to read and simple to apply without the aid of a reference library, a computer salesperson, or a niece who just graduated from M.I.T.

Reader Service No. 266

SOUND CONCEPT



©JODY THROCKMORTON 1984

FOR THE VIC 20

By A.J. Kwitowski

The Commodore VIC 20 computer has the ability to produce a wide variety of sounds. Unfortunately, this ability often goes untapped as the user is encumbered and intimidated by the assortment of POKES and timing loops necessary to produce sounds. A BASIC programmer trying to use sound to increase the quality of a program is particularly constrained, as only limited quantities and types of instructions can be executed between the beginning and end of a given sound.

DESCRIPTION

Sound Concept is a group of related programs for the bare or expanded VIC 20. The package allows easy creation and use of sounds and strings of sound within BASIC programs. Sounds created with *Sound Concept* play simultaneously with the execution of any BASIC instruction. Although usable in any type of outside program; the sounds are particularly useful in arcade-type games.

As an example, if you've ever tried to coordinate a sound that decreases in frequency with your alien's movement from the top to the bottom of the screen, I'm sure you've been disappointed. Fiddling with FOR XX TO YY: NEXT timing loops, you can achieve either smooth sound or smooth graphic action, not both. The timing loops that cause the sound's frequency to change correctly can also cause the graphic action to slow to an intermittent crawl or speed to an imperceptible blur.

Sounds created using *Sound Concept* do not suffer from this type of deficiency; they are virtually self-performing and do not interfere with graphic action. The sounds are also very easy to initiate within a user program, requiring only definition of the variable AD (ADdress of data) and branching to a subroutine. The subroutine merely POKES the address carried by AD into two memory locations and POKES one additional memory cell that starts the sound(s). Continued execution of the BASIC program is immediately returned.

A large diversity of sounds and an infinite number of sound strings can be created using *Sound Concept.* User selection of volume, frequency, tone, or change, rate of change, direction of change, duration, effects, and singular or dual channel(s) allows for highly creative effects, game melodies, and

sounds in general.

The basis for *Sound Concept* is *Sounder*. It is an interrupt-driven machine language program which interprets data and produces the sounds. Additional *Sound Concept* software consists of an *Editor* program that produces and edits sound data and a *Maker* routine that places *Sounder*, the machine language, almost anywhere in free memory.

This assortment of software makes *Sound Concept* a very useful utility that can be used in conjunction with most BASIC programs. The *Editor* is also lots of fun to play with in its own right!

Only the *Sounder* program (created with the *Maker* routine), a few statements in BASIC, and a sound data table are required for use of *Sound Concept* in an outside BASIC program. As nothing comes free, five hundred bytes must be given up to *Sounder's* machine language. Minimal additional memory is needed for the other requirements.

ENTERING SOUND CONCEPT

To use *Sound Concept*, you must first produce working versions of the *Editor* and the *Maker*. There is nothing difficult about doing this and the efforts are very well rewarded.

To function, the *Editor*, which is written in BASIC, must coexist in memory with the machine language program *Sounder*. This is readily accomplished for any of the memory configurations of the computer, as the *Maker* program automatically places *Sounder* in a specified memory area and protects it from BASIC.

For those readers with a machine language monitor, a technique will be described to join the *Editor* and *Sounder* to load as a unit. Readers without a monitor can also use the *Editor*, but must first run *Maker* to place *Sounder* in memory, then load the *Editor*.

The *Editor* should be entered into the computer exactly as given in Listing 1. If you want the program to run in the 3583 bytes available on an unexpanded VIC 20, you must be especially careful, as there is absolutely no spare memory to be used by unnecessary spaces within and between commands. When entry is completed, save the *Editor*, but do not run.

Several of the longest lines must be entered using Commodore's abbreviations for BASIC statements, e.g., POKE is abbreviated by P,shifted-O. Otherwise, the computer will not accept the lines in their entirety and the program will not run.

The *Maker* program creates versions of *Sounder* for use in outside programs and with the *Editor*. *Maker* should be entered into the computer as given in Listing 2. The data statements, lines 270 to 780, should be entered very carefuly, as any error will cause a crash when *Sounder* is activated. The program will erase itself when done, so be sure *Maker* is saved before being run.

LOADING THE EDITOR

Before using the *Editor*, provisions must be made to place *Sounder*, the machine language, in memory.

If you don't have access to a machine language monitor, such as *Vicmon* or *Hesmon*, you must load and run the *Maker* program first, before the *Editor* is loaded. If you are using memory expansion, make sure it is in place before *Maker* is loaded. When run, *Maker* will ask if you want *Sounder* placed at the top of memory or if you want to specify a starting address. Opt for placement at the top of memory, and *Maker* will automatically: POKE the machine code into a 492 byte area at the top of free memory; lower pointers to protect the code; display addresses to turn *Sounder* on and off with SYS statements; and erase itself.

Once *Sounder* is in place, the *Editor* can be load-



"Sorry, Cromwell, but the computer made a mistake, and someone has to take the blame; so we'll have to let you go..."

ed, per normal procedure. Once loaded, the addresses following the SYS statements should be double-checked and changed, if necessary, to insure that they coincide with the SYS addresses provided by *Maker*. The *Editor* will now run as designed.

With a machine language monitor, the *Editor* and *Sounder* can be combined to load as a unit. To do this, first clear the computer by turning it off, then back on. Now load the *Editor*, but do not run it. Type in direct mode: PRINT 256 * PEEK (46) + PEEK (45) (RETURN). Record the resulting value; it is the address just past the last line of the program, where the first byte of the *Sounder* machine language will be located. Now, type NEW (RETURN) and load and run the *Maker* program. When requested, input that you want to specify the starting address and enter the value obtained above. After *Maker* has completed its tasks, enter the machine language monitor, find the *Sounder* machine code, name it "EDITOR ML" and save it. Again, turn the computer off, then on, to clear it. Load the *Editor* as before. Again, don't run it. Load the machine language using the command: LOAD "CREATOR ML",1,1 for tape, or LOAD "CREATOR ML",8,1 for disk. This will return the program to its original location in memory. Now select a meaningful name for the program, something like *"EDITOR 8K"* and save it like a normal BASIC program. The computer's operating system will consider the machine language as added lines of BASIC and no alteration of pointers is necessary to protect the code or to insure a complete save. If everything went right, and it should, the united program will run as intended and can be reloaded as a unit when desired.

Please note that a version of the *Editor/Sounder* created by this technique for a particular memory configuration will crash if an attempt is made to run it on a different configuration.

EDITOR

The *Editor* is menu-driven and allows: sound creation; sound editing; playback of the current sound; playback of a string of sounds; and display of values of data for re-creating the sound in outside programs.

A major problem I had in writing *Sound Concept* was to to condense the *Editor* to the point that it and *Sounder* would both fit in a bare VIC 20. A result of this is that no REM statements could be included and the program flow is very difficult to follow. Table 1 is provided to aid in understanding the workings of the program.

The *Create* option leads the user through a series of decisions that determines the characteristics of the sound. First, volume is selected from high, medium, low, and off. Next, duration is chosen from a range of 2-254 jiffies (1 jiffy equals 1/60 second). One of nine sound modifiers can be specified: constant frequency; increasing frequency at three different speeds; decreasing frequency at three different speeds; and variations of what I call "warble" (alternating frequencies). The selection of the sound channel(s) is made from individual or paired matchings of the VIC's four sound channels. A final decision, probably, is made on the sound's starting frequency.

If the starting frequency, duration, and rate of frequency change are selected so that the sound will carry past the limits of the computer's sound channels, producing a pause, this condition is detected. The very final decision then is to leave the choice as is or change either the duration or starting frequency to end the sound exactly at the limit of the sound channels. The magnitudes of the optional frequency and duration values are displayed to aid in

the selection.

If off was selected for the volume, the *Create* routine ends after the specification of duration, as the other factors are meaningless. Pauses in a sound string are created in this manner.

On the second and each following sound creation, a decision msut be made on whether or not the upcoming sound will be included ın the previous string. Sound strings of any length are created this way.

Additional options on the main menu are PLAY SOUND and PLAY SOUND STRING. PLAY SOUND displays the chosen factors and plays the most recent creation. PLAY SOUND STRING plays any number of the created sounds in succession, up to last time the response was "yes" to the question "START OF A NEW SOUND STRING?." Both PLAY options terminate with a return to the main menu.

The EDIT option lists the chosen factors for the current sound, then asks for the factor to be changed. The options on that factor are then displayed, as in the CREATE routine. After input on the change is received, the revised sound is automatically played and a return to the main menu is executed. Reselecting the EDIT option repeats the above process, allowing a quick and comprehensive fine-tuning of the sound.

The option DISPLAY DATA provides a listing of the sound identifiers, or data, for the current sound string. The routine assumes the user's program will store sound data in the cassette buffer, as the *Editor* does, and gives the starting address of the string's data. This information needs to be recorded if the sounds are to be used in an outside program.

Although lack of memory did not allow me to incorporate a dump of the sound data to a printer, this can be readily accomplished. While the data is being displayed on the screen, press the RUN/STOP key. In direct mode, type:

```
OPEN 4,4:CMD4:GOTO122 [RETURN].
```

The information on the screen will be transferred to the printer. Again, hit RUN/STOP. Now type:

```
PRINT#4:CLOSE 4:GOTO122 [RETURN].
```

Program execution will resume with no loss of the created sounds.

SOUND DATA

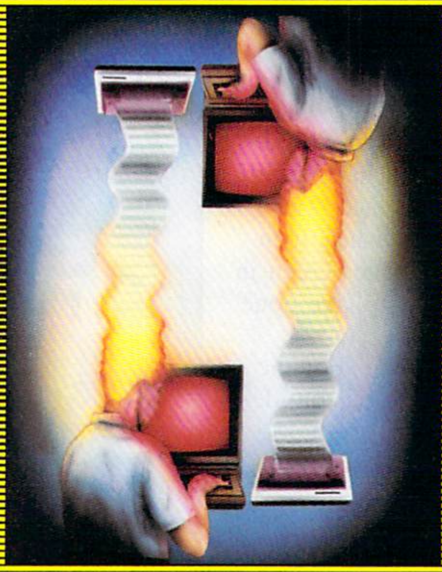
The structure of *Sound Concept* provides for very flexible storage of data in memory. Each sound requires three bytes of memory for storage of the sound data. Any string of sound is merely a succession of sounds/sound data bytes, where the last data byte of the previous sound is altered, signifying *Sounder* to continue with the next sound. The only input *Sound Concept*'s machine language requires is the address of the first byte of sound data; all the rest is taken care of automatically. Therefore, the user is free to put data for sounds/sound strings anywhere memory is free, i.e., not occupied by BASIC or *Sound Concept*'s machine language. Placement of sound data at several different memory areas is also feasible.

The one restriction on data placement in normally free memory is the first 16 bytes of the cassette buffer, 828 to 845 decimal, \$033C to \$034D hex.

Continued on page 96

TABLE 1

Line 2	Define screen color variable; set data table to cassette buffer.
Line 4	Subroutine to print highlighted numbers.
Line 6-10	Subroutine to clear screen; change border color; print options.
Line 12	Subroutine to change letters to numerical values.
Line 14	Subroutine to clear screen, position cursor, and initiate highlighted printing.
Line 15	Subroutine to read text.
Line 18	PLAY SOUND STRING subroutine.
Lines 20-22	CREATE SOUND main subroutine.
Line 24	Clear sound initiation cells; read options data; start SOUNDER.
Lines 26-30	Main menu.
Lines 32-36	Subroutine to select channel.
Lines 38-42	Subroutine to select starting frequency.
Lines 44-48	Subroutine to select duration.
Lines 50-54	Subroutine to select sound modifier.
Lines 56-84	Subroutine to detect, display, and change pause.
Lines 86-90	Subroutine to select volume.
Lines 92-102	Subroutine to zero factors and determine sound string continuation.
Lines 104-106	EDIT subroutine.
Lines 108-120	PLAY SOUND subroutine.
Lines 122-128	DISPLAY DATA subroutine.
Lines 130-136	Main menu subroutine.
Line 138	Turn off sounder.
Lines 140-148	Data



CODE GENERATING PROGRAMS

By Bernhardt Hurwood

"I don't want to know anything about programming a computer. I just want to use one."

The preceding statement, while sacrilege to the average *Ahoy!* reader, represents the attitude of the majority of microcomputer owners. Most lack either the time or the inclination to delve deeply into programming language. Neither, however, can the average user afford all the software he would ideally like to own; nor, in many cases, can the software which fits his unique needs be found anywhere.

If you've read the introduction to program generators that begins on page 23, you know that a solution exists. So let's examine the *CodeWriter* series, which has made it possible for non-programmers, using plain English, to create their own software for whatever purpose they choose.

CodeWriter Corporation (formerly Dynatech Microsoftware) offers two file writing programs, *Home FileWriter* and *FileWriter 2. Home FileWriter* ($69.95) is the less expensive and simpler. Typical of its applications are tax planning, recipe files, budgeting, scheduling of bill payments, household inventories, and calendar planning.

FileWriter 2 ($99.00) is more sophisticated. It was formerly named CodeWriter—now the name of the entire series. And like the other program generators in the series, it provides all the necessary on-screen instruction and prompting to permit non-programmers to design the applications they need in minutes. One can build a whole library of unique programs; furthermore, *FileWriter 2* automatically debugs programs, enabling trouble-free operations in the future. It even has a menu-creation capability which allows programs to be run with a single keystroke. Business applications include accounts payable, accounts receivable, inventory control, personnel records, ledger systems, and sales forecasts.

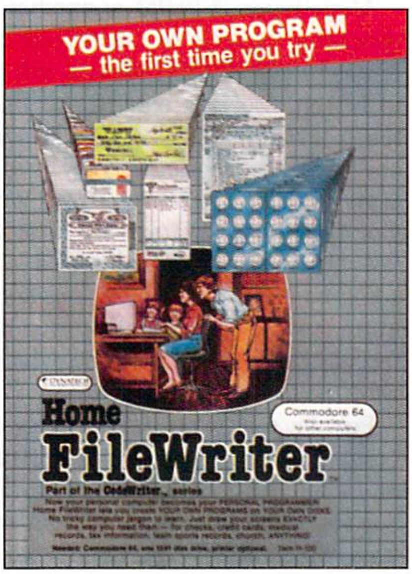
MenuWriter ($29.95) is another practical program generator, capable of creating as many as twenty different screen menus, each of which will display up to fifteen directory entries. All the user has to do is touch an individual key and *MenuWriter* will load and run any program displayed in the menus. Says Dynatech's V.P. of Sales, C.W. Wright, "The user simply tells *MenuWriter* what he wants to do and how the files are to be indexed—letters, numbers, or some combination—and *MenuWriter* automatically translates the plain English instructions and writes the computer language code which actually does the work in the computer."

For the user who needs highly specialized programs that are too tied into personal needs, the *ELF*, or *Easy Language Form* system ($49.00) is ideal. Suppose you want to build your own ultralight aircraft which will be powered with a VW engine modified to run on kerosene. Before you get going you

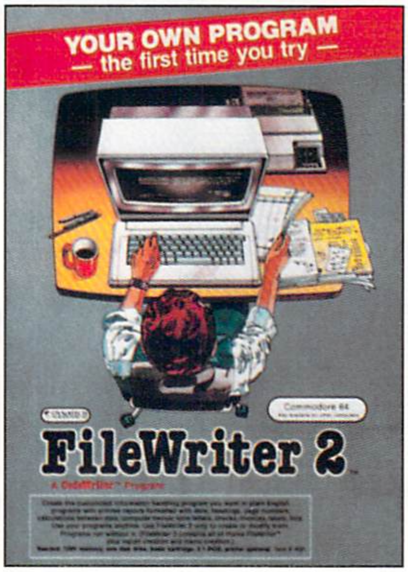
must calculate all the necessary aerodynamic elements, then your projected costs, and finally what your operational and maintenance expenses will be. With *ELF* you can do all of that without having to understand an iota of programming language. The custom programming process is purely visual. It involves the entry of your own ideas on the screen in a simple basic language form. Once you have done that, *ELF* writes the computer code and your uniquely created program is ready to run. For the business user, *ELF* makes it possible to custom-design form letters, personal spreadsheets, even analysis systems for organizational purposes. For the recreational user, it makes possible the creation of games.

At the same price as *ELF*, the computer novice can now have *Dialog*, which enables him to create his own interactive programs. This is especially valuable to those who want to write educational or game programs. With *Dialog*, a user controls text, questions, and answers, all by means of simple onscreen procedures. Before *Dialog*, interactive programming could be performed only by an experienced, professional programmer.

For the game addict, *AdventureWriter* ($49.00) is a must. Imagine the possibilities of transferring your own adventure fantasies to the screen via the home computer. Is science fiction your bag? If so, you can dream up any plot you wish. Send your hero across the galaxy in a faster-than-light starship to save the earth from destruction. The enemy can consist of scaly, super-intelligent, carbon monoxide-breathing creatures that can beam deadly lasers at any target. It is their plan to invade earth and turn it into a planetwide creche for their roelike embryos. They



Home FileWriter ***lets you generate programs for budgeting, tax planning, payment scheduling, etc. READER SERVICE NO. 230***



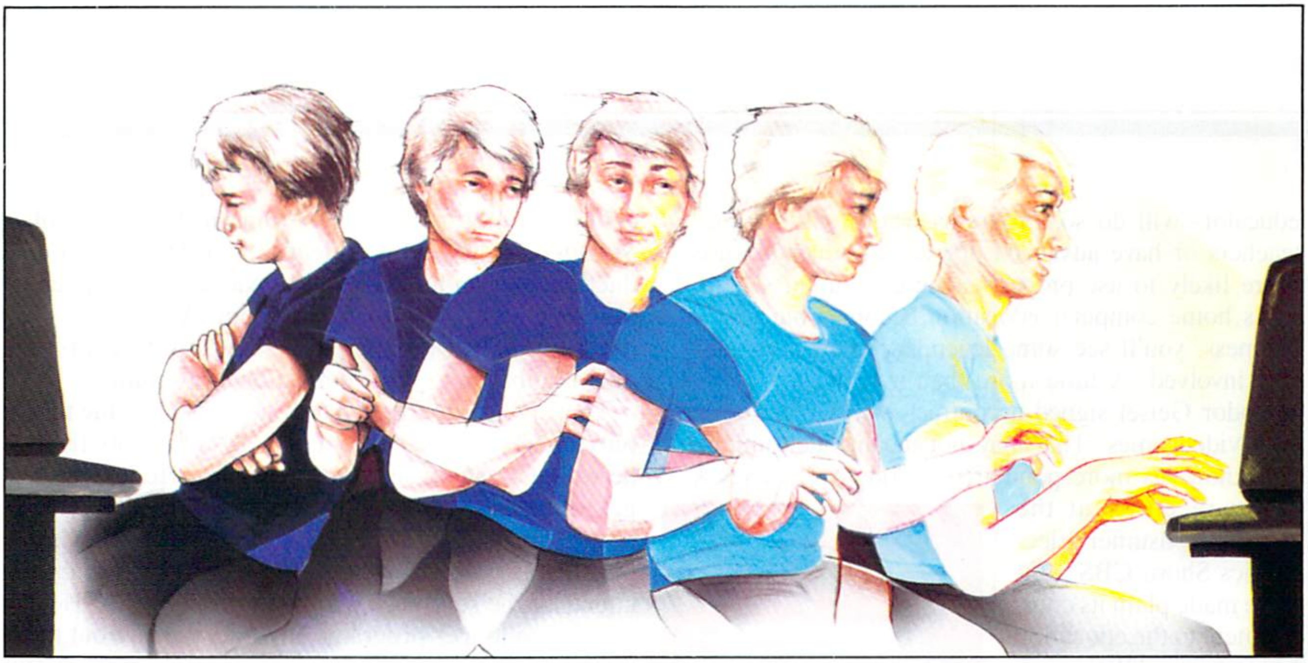
FileWriter 2 ***features automatic debugging and a menu-creation capability that allows programs to be run with a single keystroke. READER SERVICE NO. 231***

must first burn off the earth's atmosphere and replace it with the lethal carbon monoxide they require. It is the hero's task to thwart the monster race and destroy them.

If that sort of thing inspires you to sit back and yawn, perhaps you would prefer to create an alternate world where magic reigns instead of science; a world of sorcerers, dragons, flying horses, magic swords, witches, unicorns, and hidden treasure. The characters you people this world with depend strictly on you. Think of the hazards you can create: boiling seas, caves inhabited by cannibalistic amazons, poisonous fogs, and vampires' castles.

But if fantasy and science fiction are not your cup of tea, you can create jungle adventures, war games, undersea exploration games, traditional treasure hunts, even adventures in the wilds of the corporate boardroom. As Warren Shore, President of Codewriter, put it: "You are not buying a one-purpose game application. You are getting a games system that allows you to create a virtually limitless number of games."

An added feature, which technically applies to all of the program generators described here, is that any program you create is your own property which under existing laws may be copyrighted in your name, and which then becomes proprietary software for the rest of your life plus fifty years. For the individual with a creative mind and an entrepreneurial bent, this could easily become the beginning of a new career, or at the very least, a profitable sideline. But whether you want to start a software business, or merely take advantage of your computer to its fullest capability, program generators such as these open up a whole new world for users. □


© Perry A. Realo 1984

EDUCATIONAL SOFTWARE: A GUIDE FOR PARENTS

Part IV

By Richard Herring

The Office of Technical Assessment, created by Congress to keep it informed of technical usage, predicts that half the families in this country may have home computers by the end of the decade. Of the families that own computers today, 46 percent bought them with the idea of improving their children's education. What are the brands to buy? During 1983, 87 percent of the computers bought by schools were Apples, Radio Shacks, and Commodores, according to a study by Knowledge Industry Publications of White Plains, New York, *Microcomputer Hardware and Software in the El-Hi Market 1983-87.*

Terrel Bell, Secretary of the Department of Education, when interviewed by InfoWorld said, "I believe that the computer, especially the inexpensive microcomputer, is going to change American education probably more dramatically than any other development that I see on the horizon." Bell made it clear that he thinks micros will "be a great asset in providing students with an interactive learning environment." This series of articles is written to help you turn your Commodore into an interactive learning environment for your child to use at home.

EDUCATIONAL VALUE

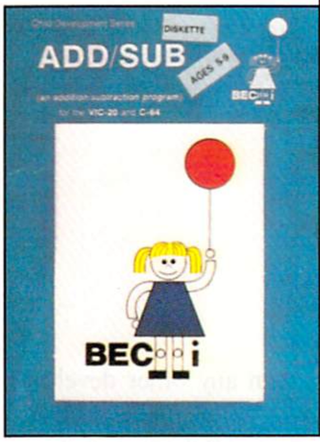
The criteria by which you, as a parent, can judge the educational value of software are both the hardest and the easiest to apply. The packaging and documentation should provide some of the information you need. Look for the background and education of the program's authors. Certainly a high school kid who is a programming whiz may write good educational software, but it is more likely that professional

educators will do so. Programs whose authors are teachers or have advanced degrees in education are more likely to use proper teaching techniques.

As home computer education becomes big business, you'll see some legendary characters getting involved. A little more than a year ago, Theodor Geisel signed a contract with Coleco to design videogames. That may not mean much unless you know his more popular pseudonym—Dr. Seuss. More recently, at the Winter Consumer Electronics Show, CBS Software made plain its commitment to the education business. CBS announced a line of educational software, based on the television series *Mister Rogers' Neighborhood,* on which Fred Rogers will collaborate. Already in the big name business, CBS has worked with the Children's Television Workshop (Sesame Street).



Match timing with skill. READER SVC. NO. 234



Child picks time limit. READER SVC. NO. 235

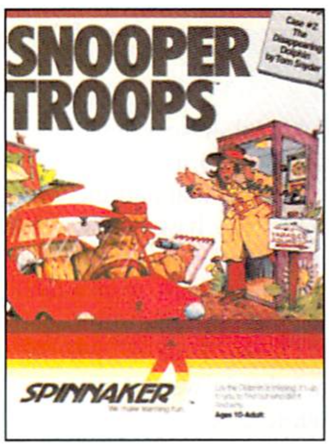
Also, look for a collaboration of authors. To expect a single author to be an educator, programmer, musician, and graphics designer is asking a lot. David Seuss, president of Spinnaker Software, says it is not uncommon to have professional musicians involved in developing educational software. And he's right. In mid-1983, Atari hired Ed Borgas as a musical consultant. Borgas has composed scores for movies, commercials, and TV shows including *Peanuts* specials, *Here Comes Garfield,* and *Sesame Street.* Educational music software for Commodore computers from CBS has been produced by the Emmy-award winning Dovetail Group. This software features the Jazz Scats, an offbeat musical trio.

Other educational software authors lay claim to their reputations based on what they have done with computers, not books or TV. Tom Snyder, author of *Snooper Troops* and president of Tom Snyder Productions, began his career as a teacher who happened to have a computer in his classroom. As a self-described nerd who wanted to play with the computer, he began to write educational programs.

That's when he found that if he wrote a game that only interested one child in his class of twenty-five, he would create twenty-four problems. He learned to get lots of kids involved and to avoid the general trend toward boring educational software. Snyder now lets his creative juices flow, but through the programming talents of the seventeen people in his company. He says he no longer programs anything they let out the door.

Other, less well-known teachers have turned into programmers too. Dale Disharoon is a good example. Just a few years ago he was teaching at an elementary school in California. He is now the author (along with Jim Bach) of Spinnaker's *Hey Diddle Diddle* and *Alphabet Zoo.* In *Hey Diddle Diddle* a child must unscramble the lines of a poem (there are 30 poems with 8 lines each). When he is successful, his reward is a song and a colorful graphics display of the characters in the poem. This program reinforces reading skills, grammar, rhyming, and logical thinking.

Next, consider the credentials of the company. If a software company has been around for a while and has a history of well-received products, there is probably a reason. Big name companies are less likely to attach their reputations (and their advertising budgets) to marginal programs. That does not mean you should rule out new, single-product companies, just that you should evaluate the software differently. Once a software company realizes the traits that underlie good educational programs, the results are obvious. Spinnaker's Seuss demands that educational software be innovative. Seeing only three themes in arcade software—shoot something, chase something, or dodge something—he wants more from his company's products.



Snooper Troops ***teaches data organizing. READER SVC. NO. 236***

Is the program clear about its specific teaching objectives? The packaging should not only tell you what those objectives are, but when you load the program, the objectives also should fit what you see. If they do not, try to figure out what the program is attempting to do. The objectives should be geared toward the computer as a tool. If meeting these objectives can be accomplished more easily or better with a different tool, then use it, not a computer.

In the pair of *Snooper Troops* adventure games—*The Granite Point Ghost* and *The Disappearing Dolphin*—the educational intent is clear. It is to teach children to organize data; they must draw maps, take notes, and analyse facts. These games meet their objective. Their author, Tom Snyder, says that he likes the fact that the instructions do not say to "Get a pencil, get paper." As kids play the game, though, they realize they cannot remember all they need to know and soon they are scouring the house for a pencil.

Spinnaker's *Story Machine* is designed to let children write their own simple short stories and then automatically animate them onscreen. However, it falls short of its objectives. Some real limits are placed on the child's creative ability. If you try to use too many actors (nouns) on the screen at one time, the program will stop. Or, the program may cross out a word you have just typed in. Before your story gets very complicated, you are liable to run out of space. Once you figure out the boundaries of your creative environment everything works fine, but those boundaries do not encompass as much territory as the program objectives might lead you to believe.

Many well thought out programs even include ways to determine if the child is actually learning what the software purports to teach. These tend to be some of the newer educational titles since much of the early software did not teach—it just let you practice what you already knew. *Preschool IQ Builder I* from PDI lets you check your child's score on any lesson. Just press the zero key and you will find out if your child has mastered the material.

Make sure the educational program you're about to purchase is not a poor quality game, which would never sell well, repackaged as educational software in hopes that its slow action will be acceptable. "Few educational titles are based on rigorous academic protocols," according to Sherwin A. Steffin, an educational technologist with expertise in high school and college course design and vice president of Eduware. "There is great stress on being colorful and entertaining. But often the instructional values are missing. Some products simply rehash software designed for fun rather than for learning." If the program does not say what its teaching objectives are, perhaps the author did not know.

Does the program list the prerequisite skills needed by the child? Even very young children can grasp advanced concepts if they are presented properly. By the same token, a piece of educational software geared toward your child's age group may require a skill that your child does not have yet. Unless you are willing to spend a substantial amount of time with your child and the program, frustration is the sure result. Programs should give you some idea of the behaviors and knowledge your child needs to use them successfully.

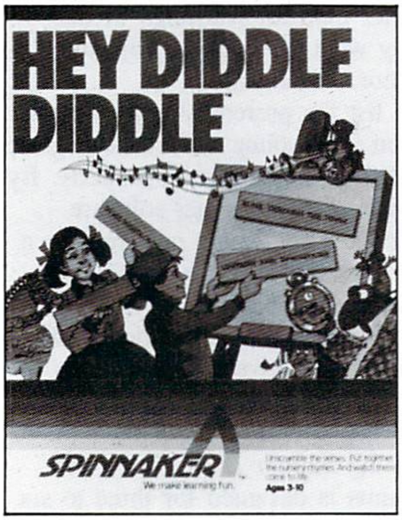
Back to *Preschool IQ Builder I,* we find a program which PDI claims is designed for three to six year olds. Teaching the concepts of same and different, the program shows two objects on the TV screen. The child must decide whether the colors or letters are the same, or match letters on the screen with those on the keyboard. For many six year old children this activity will be much too easy. You should decide its value for your child before you buy. Your three year old will be able to use it over a period of years. Your precocious five or six year old may look at it only once.

Watch how the program responds when you type in something. Do the graphics draw your attention to irrelevant portions of the screen? Your child's eyes will certainly be drawn to the action. Sandra Curtis, director of research for Joyce Hakansson Associates, points out that children tend to learn what they find interesting, not what adults say is important.

Some software disrupts your attention to the learning objectives by its own attempt to keep your attention. *Typing Strategy,* from Behavioral Engineering, is an elaborate tutorial for beginning or advanced typists. But its busy screen is distracting. Two, or even three, things are usually going on at one time. How do you decide what to pay attention to?

The action should be where the learning is. *The Game Show* is a good example of a program which draws your attention to relevant facts. Advanced Ideas' program is similar to the TV game Password, with the moderator and one of each team's players shown on the computer screen. When your computer partner gives you a clue, the moderator looks at your partner, the clue appears in a bubble above his head, and all other action on the screen stops. As the clue to the secret word is given, the action directs your attention to that clue, then stops when you are looking right where you should be.

Does the computer respond with a squawk when



In **Hey Diddle Diddle,** *the child must unscramble the lines of one of 30 eight-line poems. The program reinforces reading skills, grammar, rhyming, and logical thinking.* *READER SERVICE NO. 260*

you guess wrong? Some children are very hesitant to use software which lets everyone in the room know when they give an incorrect answer. This is less of a problem with Commodore computers than with Apples and Ataris. Many programs for those machines generate sounds which can not be turned down or off merely by twisting a TV knob. Watch your child's reaction to the various beeps and squeeks used by programs. Even a beep for a correct answer, if it is the same beep every time, can get old.

Does the program present problems in traditional ways? For younger children, arithmetic problems should be presented vertically. Only older children will be familiar with the horizontal display of math problems most common on computers. Symbols should match those used by children. To most kids, an asterisk does not imply multiplication.

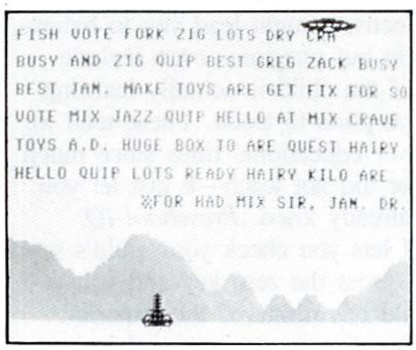
ADD/SUB from Boston Educational Computing Inc. is a good example of familiar problem formatting. Choosing addition or subtraction, you can have the program present problems with one to four digits and with decimals. Problems that require carrying and borrowing can also be included. Large, easy to read numbers appear onscreen with a plus or minus sign and a line drawn between the problem and the answer. Answers are given one digit at a time, from right to left, just like they would be on paper. If a kid is just learning to carry or borrow, he can get help by having an arrow point to the column from which he has borrowed or to which he must carry. This type of problem format will seem familiar to your child; he can concentrate on the arithmetic, not new rules imposed by the software for doing it.



Story Machine *lets children write their own simple short stories and automatically animate them onscreen.* *READER SERVICE NO. 261*

Similarly, for multiple choice questions, the answers should be stacked one above the other as they appear in most standardized tests. The child's choice should not be made by highlighting the whole answer he wants to pick. Rather, just the number next to the answer should be highlighted since this is visually more similar to circling the correct answer on a written multiple-choice exam.

Another example of using traditional methods is Academy Software's *Typing Tutor.* It comes with the same kinds of instructions that a typing instructor would give. Finger placement, posture, and elbow position are all described. Punctuation must be followed by proper spacing. And, unlike some other typing programs, the scoring formula used is the same one you would find in any beginning class for budding young typists.

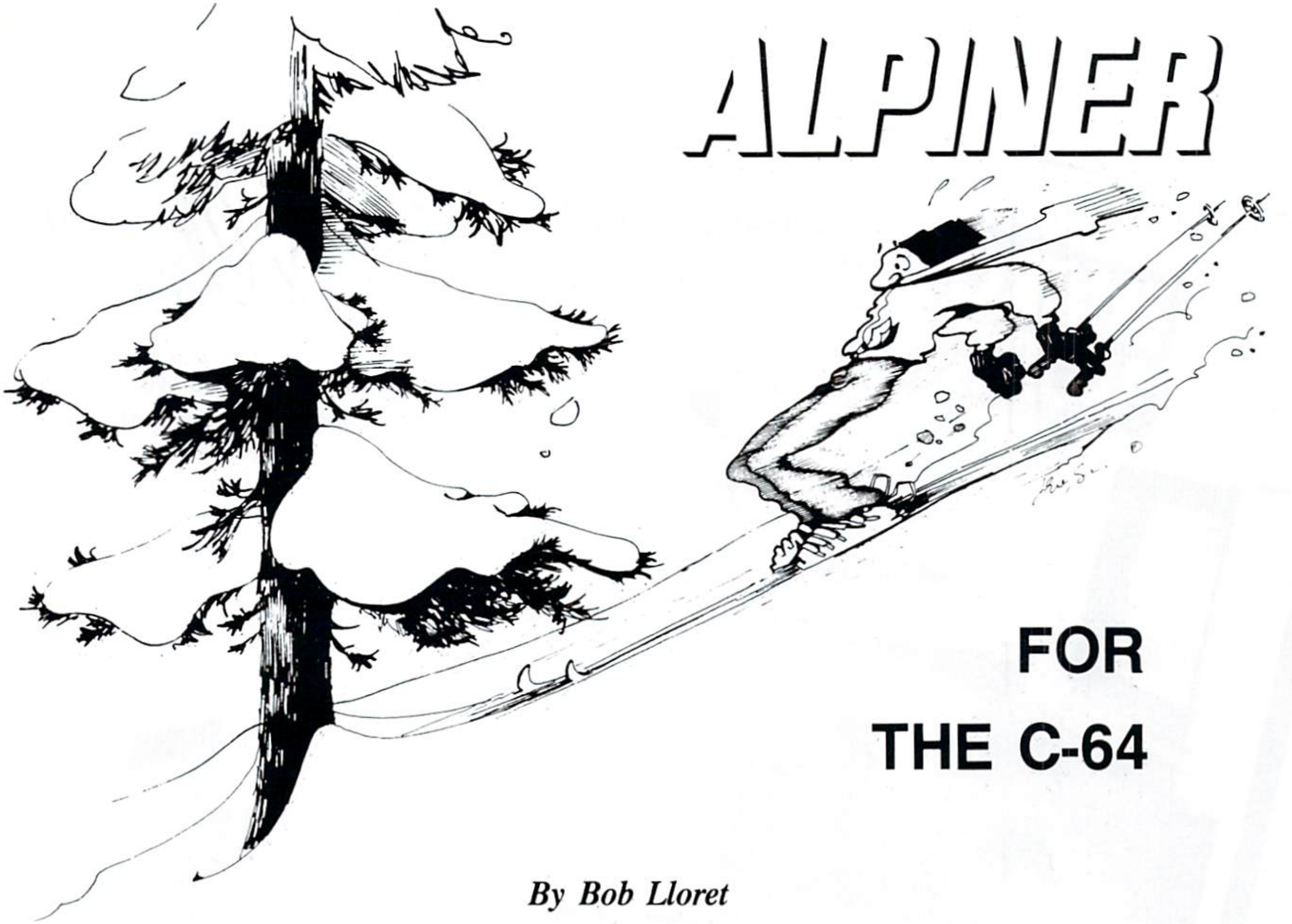


Typing Tutor *uses many of the same instructions that a typing instructor would give.* *READER SERVICE NO. 262*

For many types of educational software, timing is also important. Whether in school at age six or in the business world at age thirty-six, your child will eventually face time limits. *ADD/SUB* lets you choose the speed at which answers must be given. With a default of about seven seconds, this speed option should allow your child to benefit from the program over a longer period of time.

Another game from the same company, *NUMBER-*

Continued on page 91



ALPINER

FOR THE C-64

By Bob Lloret

With the winter ski season over for most of North America, and many ski bums wavering between a trip to the Alps and suicide, *Ahoy!* provides an alternative—*Alpiner* for the C-64.

Alpiner is not just another skiing program, it's a challenge that only the best will conquer. Designed for one to four players, it has three different screens (called legs) and three difficulty levels. I suggest you start with the easy level to get the feel of the courses. Type in the program, then save a copy to disk or tape in case of any typing errors. Since the program moves the character set and uses an assembly subroutine to read the joystick, a typo may lock up your computer.

After saving a copy of the program, type "RUN" and press return. The fun begins when the title screen appears and the question "ENTER NUMBER OF PLAYERS" is displayed. The program will accept from 1 to 4 players. Less than 1 or more than 4 will be rejected and the program will restart. After the number of players has been entered, the program will ask for the first names of the players. Enter the first names and press return. The next screen will ask you for the difficulty level. Select 1 for Amateur, 2 for Professional, or 3 for Alpiner level of play and press return. The message "Hold on. . .I'm arranging the course" will appear on the screen with a short pause. In this time, the program is moving the character set, creating sprites, and installing a machine language subroutine in memory to read the joystick movement.

After this is completed, the program will show the start of the first of three legs with the message "GET READY" and the name of the first player. A short delay and the screen will start scrolling upwards as you ski down the course. The object is to ski between the two flags (also called Gates). You will be penalized for every flag or obstacle your skier hits on the way down. When you hit an obstacle, your skier will lose his balance and go up on one ski but then recover. At the end of each leg, your totals will be displayed with the message "NEXT LEG COMING UP" at the bottom of the screen. You will then go into the second leg which is a little more difficult and then the third leg which is still more difficult. Your final totals will then be displayed with the message "NEXT PLAYER PLEASE" on the bottom of the screen. Player Number 2 will then start from the beginning of the course and go through all three legs with his totals displayed after each leg. This will continue until all players have gone through the whole course. All players' final totals will be displayed and the program will end.

For those interested in programming, the following is a list of program flow:

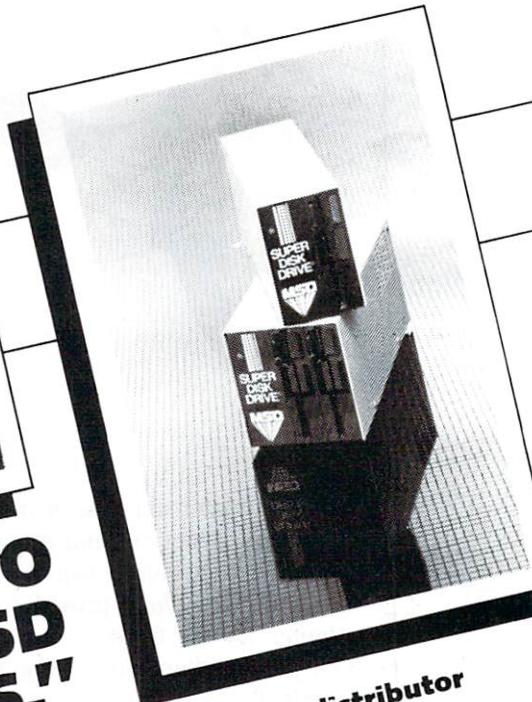
Lines 1-4	Title and author information.
Lines 10-70	Title screen and user information.
Lines 100-180	Player difficulty level.

Continued on page 93



[signature]

Lee Wiltrout
Micro Worx, Inc.
developers of Prosys Software



"IT IS EASY TO GET SPOILED BY MSD DUAL DISK DRIVES."

Dealer and distributor inquiries invited.

"We love MSD's new dual disk drive, the SUPER DISK II. The programmers at our Lubbock store run SUPER DISKS continually for weeks, and they never overheat! This is important to us because we develop software for the Commodore and the time we save returns as profits.

"MSD's highly-reliable single and dual disk drives are at the very top of our list of recommended products. Our sales have dramatically increased since we brought MSD products into our stores.

"Sharon Bray, Micro Worx vice president, and I agree that MSD products help us provide our customers with the very best peripherals for expanding their Commodores into the real world of serious computing.

"We use MSD products and that's enough proof for our customers!"

Lee Wiltrout, Manager
Micro Worx, Inc., Hurst/Lubbock, Texas

Commodore owners, now you can gain access to expanded capability for your computer. SUPER DISK II will format, copy, and verify in less than 2 minutes, a procedure which normally takes 30-45.

Expand into the real world of serious computing with MSD products. Call to- day for the nearest dealer of the reliable . . . available alternative.

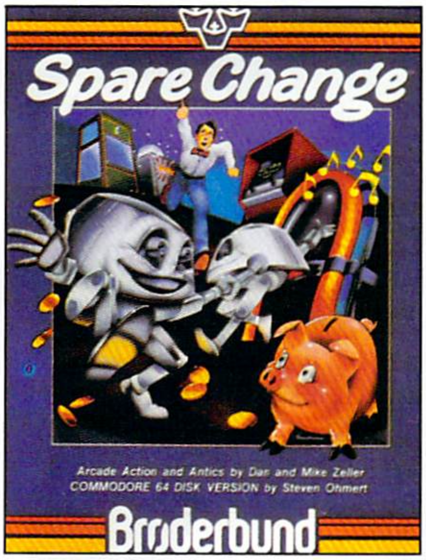
MSD SYSTEMS, INC.

10031 Monroe, Suite 206, Dallas, Texas 75229 • 214/257-4434
Outside Texas 1-800-527-5285

Reader Service No. 275

SPARE CHANGE
Broderbund Software
C-64
Disk

There's trouble at the Spare Change Arcade. Two Zerks have escaped from their coin-operated Zerk Show machine and are on the loose. They're trying to collect enough tokens to retire but, as owner of the arcade, it's up to you to stop them.



Keep the Zerks gainfully employed.
READER SERVICE NO. 232

To keep them from retiring you must deposit eighteen tokens in the arcade's token bins before the Zerks can get five tokens into their piggy bank. Tokens can be taken from the token machines that are around the arcade and, occasionally, you can find them in the coin-return slots of the payphones.

In order to keep the Zerks from filling their piggy bank before you can fill the token bins, you must provide distractions to keep them from the task at hand. Fortunately, the Zerks aren't terribly bright, and are easily distracted.

One way of distracting the Zerks is by means of the juke box. Simply drop in a token and when the music starts, the Zerks will drop whatever they're doing and begin dancing like fools. As the game progresses, you can keep them busy by depositing tokens in the popcorn machine (Zerks love to watch the stuff popping), or by calling one payphone from another. The Zerks will answer and take time out for a little conversation.

Once you've acquired at least ten tokens and placed them in the bins, the door to the Zerk show opens and you can enter for a brief rest and to get credit for tokens already collected. For every token above nine, you get a money bag. Collect nine money bags and you advance to the next level. Every time you complete a level, you get to watch a humorous Zerk Cartoon and after completing four levels you get your own Zerk Show booth for your collection.

Spare Change is a mildly addicting game with a sense of humor. Although veteran gamers will quickly find it easy to outwit the Zerks repeatedly, level after level, the game includes a "Control Zerks" option that allows you to increase the game's difficulty. By pressing f1, you call up the Zerk Control Panel that lets you modify nine facets of the Zerks' behavior. This option insures that your hundredth visit to the Spare Change Arcade will be as much fun as your first.

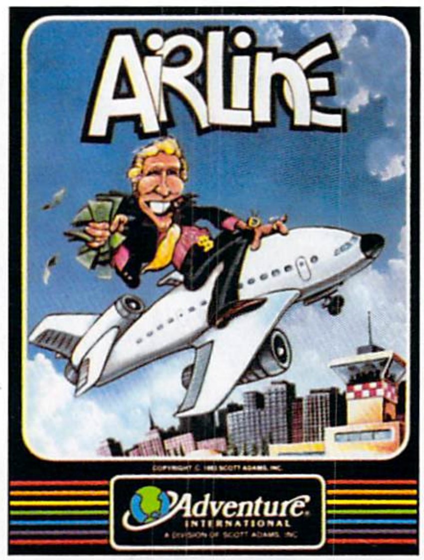
Broderbund Software, 17 Paul Drive, San Rafael, CA 94903 (415-479-1170).

—Lloyd Davies

AIRLINE
Adventure International
C-64
Disk

Because *Airline* makes little use of the C-64's graphic capabilities and produces not so much as a single beep from the computer's sound chip, you might consider it a board game that you play on your computer. Like many board games, *Airline* is a strategic contest which involves the conquest and management of territory.

The territories, in this case, are thirty-six U.S. cities like New York, Dallas, L.A., and Washington. Each city is acquired not



A corporate battle for the skies.
READER SERVICE NO. 233

through military confrontation and the placement of troops, but by purchasing landing rights and basing commercial aircraft at each location.

Up to four human or computer-controlled players begin the game with $100,000 each. This initial capital is used to buy landing rights which vary in price directly with the size of the city. While landing rights in New York will cost you almost twice as much as L.A. rights, once the fares start rolling in, a New York-based plane will earn far more than a similar plane based in L.A.

A second factor in determining the amount of the fares you will receive is the type of aircraft in



operation. When you purchase a plane to base at one of the cities to which you own landing rights, you can chose from a prop, airbus, DC-10, B-747, or Concorde. The larger the plane, the greater the revenue, and the costlier the original investment.

Each time all players complete their turns by either buying landing rights or aircraft, the computer calculates the fares for each player, and adds these to the scores. At the end of the game the airline entrepreneur with the most money wins.

Airline is an enjoyable game, but because your options are limited to buying unowned landing rights and purchasing airplanes to base at those sites, the game often grinds to a halt if you choose to play a large number of turns. Once all landing rights have been bought and you've upgraded all your planes to Concordes, there's nothing left to do but collect the computer-assigned fares. What the game really needs is a larger assortment of cities, more interaction between players, and an element of chance.

Airline could have had players competing for landing rights in hundreds of cities all over the globe. Players could have been allowed to trade or even auction off landing rights and aircraft to other players. Individual fares could have been selected by the players with the computer determining how much revenue that particular fare would generate. Finally, fares could vary from time to time because of seasonal travel, weather, natural disasters, etc.

Instead, *Airline* is a simple contest which gives players little to think about and even less to do. With *Airline*, the skies may be "friendly," but they're also pretty boring.

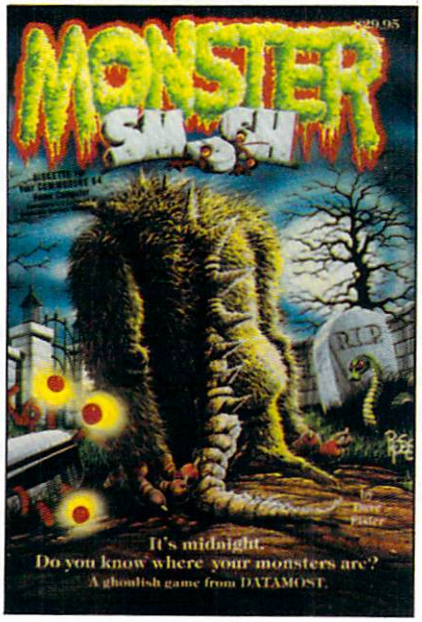
Adventure International, Box 3435, Longwood, FL 32750 (305-862-6917).

—Lloyd Davies

MONSTER SMASH

Datamost
C-64
Disk, cassette; joystick

It isn't often that a unique skill-and-action game comes along. *Monster Smash* is one.



Monster Smash: *a graveyard mash?*
READER SERVICE NO. 245

You are the caretaker of a monster-infested graveyard. The monsters ("A wild and furious bunch who like to party and raise the dead") are dying to escape the graveyard and wreck havoc on the nearby town. Most other caretakers might let them go—good riddance, no?—but you are a community-minded sort, so your task is to stomp them as they go racing by.

The game screen can roughly be compared to the *Hollywood Squares* board. A grid is formed by a series of horizontal gates. Forming the squares are tombstones with which you flatten the creatures. The creatures appear onscreen upper left and make a dash for screen right. With your joystick you flip the gates into a vertical position and down again to a horizontal. In this manner you divert the monsters downscreen and into the path of your tombstones; they will scurry to the right whenever a gate is open to them. As the monsters keep coming, three and four and more at a time, you keep slamming and swishing them by pushing the fire button. Complicating matters are visitors and kids. If you allow them to pass through the graveyard unsmashed, you gain extra points. Keyboard control options allow more precise control of gates and tombstones, and there are six levels of difficulty.

Don't be fooled by the title. There is very little in the way of spooky atmosphere, no creepy music, screams, or digital gore. The monsters are funny-looking and lovable, really, with names like Spirits, Smokey, Eyes, Claws, Eggs and Snakes. The sound effects are like staccato broom-strokes; they help contribute to the helter-skelter nature of the game. And it *does* get chaotic, with dozens of gates and tombstones flipping and an endless wave of scurrying groaties rushing by. Mindless but witty reflex-testing fun.

Datamost, 8943 Fullbright Ave., Chatsworth, CA 91311-2750.

—Tim Moriarty

THE SMART 64 TERMINAL PLUS 2

Microtechnic Solutions Inc.
C-64
Disk

This full-featured terminal package is designed to be used in conjunction with the Commodore 64, one or two disk drives, a

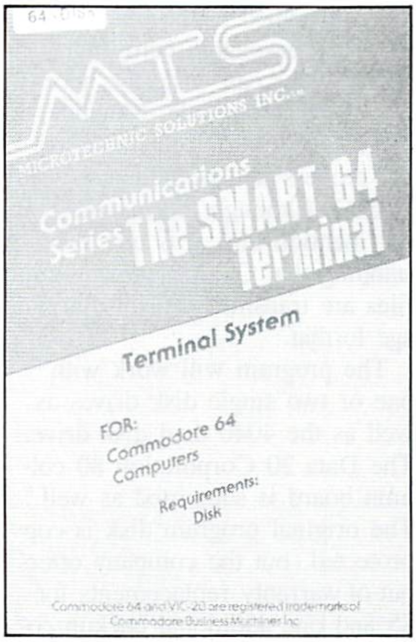
printer, and either the model 1600 modem or the model 1650 automatic modem. It's packed with useful features that greatly assist the user in performing various telecommunications chores, not the least notable of which is the ability to upload and download files.

Operation of the program requires two disks, the original program or "System" disk, and a user disk. The latter is initially prepared using the *Smart 64 Build* program included on the original program disk. The user disk contains the various parameter files such as the ASCII conversion tables, modem settings, function key assignments, printer parameters, and screen color settings. All download and upload files are stored on this disk as well. If you have occasion to communicate with more than one host computer, you should create a separate customized user disk for each one.

The package provides a number of convenience features. The function keys are put to good use with f1 transmitting the user ID and f2 the user Password. The 28 kilobyte buffer, used for temporary storage of downloaded data, is toggled on and off with function key f3. Up to four 80 character lines may be assigned to function keys f4-f7 for single key transmission to a remote terminal. All standard control codes are transmitted with the CTRL key. The main option menu can be accessed at any time with function key f8. A screen dump to the printer is available while on-line by simultaneously pressing the Commodore/British Pound keys. Communication with the host computer is automatically suspended when the last two options are exercised.

The ability to suspend communication and send stored data to disk is provided for as well. A somewhat abbreviated DOS wedge utility allows disk access without leaving the main program.

The feature which will be of greatest interest to most users is the ability to download and upload files. The *Smart 64 Terminal Plus 2* has several ways to handle these operations. To begin with, all upload and download files must be ASCII files. These can be either in Commodore ASCII



A full-featured terminal package. READER SERVICE NO. 254

format or in standard ASCII format. The program will handle the conversion from Commodore ASCII as required. The use of ASCII text files is necessitated by the industry communications standards. Binary or program files will invariably contain characters that will be interpreted by the terminal software as various control codes. Any attempt to transmit a binary file in its original form will lead to unpredictable results.

Several utilities are provided to get around these restrictions. Two programs handle the conversion for BASIC program files to sequential text files and the reverse conversion procedure. Only standard VIC/64 BASIC commands without extensions are converted. Special commands, such as from the VIC Super Expander or from *Simons' BASIC*, will be saved as direct text.

This last limitation can easily be overcome on downloaded files by loading the program, listing the problem lines to the screen, and hitting a carriage return over each problem line. The conversion from program to sequential files may be a problem with most BASIC extensions. In this case, the computer itself can be used to create the sequential file. This is readily managed by OPENing a sequential file in direct mode, executing a CMD command to the open file and LISTing the program to the disk drive. Do not forget to complete the procedure with a PRINT# followed by a CLOSE command. The resulting text file can be uploaded by the *Smart 64 Terminal Plus 2* without any difficulty.

To aid in the processing of downloaded files, a review and extract feature is provided. This makes it easy to separate program text and image files from the body of the download files for further processing. A word processor which reads sequential files can be used to perform this function as well. The limiting factor is the size of the download file. The *Smart 64 Terminal Plus 2* can create a text file which will fill all available space on the disk. Most word processors are limited to available memory in the size of the files they can handle.

Messages and other text must also be processed from sequential text files. Brief text files, up to nine 80 character lines, can be created and edited by the 1650

Automodem section of the *Smart Terminal* program. Larger text files will have to be created with your own word processor or text editor. These files must be created as sequential files. *Easy Script* is suitable for this purpose. Just stick to plain text with carriage returns, avoiding any of the embedded format commands, to stay out of trouble.

An alternative method is provided for handling program files. This approach will handle machine language programs as well as all extensions to BASIC. The process involves forming an ASCII image file of the original binary program file. What this does is read the original program data, a byte at a time. The corresponding hexadecimal values, in ASCII format, are then sent to a sequential text file. This file can be easily uploaded and downloaded as any other text file. The reverse conversion involves reading the ASCII file, converting the hexadecimal representation back to the corresponding binary value, and sending the corresponding values back to a program file.

The procedure worked well. The only limitation is that both the sending and receiving parties must have the necessary conversion software. The file format is fully described in the manual so that any reasonably competent programmer can write the necessary conversion code. Machine language would be desirable as the process will be rather slow in BASIC. Although the manual does not mention this, it is our understanding that Compuserve program files are transmitted using this image format.

The program will work with one or two single disk drives as well as the 4040 dual disk drive. The Data 20 Corporation 80 column board is supported as well. The original program disk is copy protected, but the company offers out-of-warranty replacements for $5 and one additional backup copy for $10. This copy protection will be a slight nuisance for single disk drive users as it requires swapping of the user disk with the program disk whenever major operating modes are changed (such as when switching from on-line mode to file operations mode). The bad sectors used for copy protection were also checked at every major mode change. This last procedure should really be unnecessary, after the initial start-up, in particular as the resulting head chatter puts additional strain on the disk drive components. It is something of a shame that the company felt the need to add this to their disks, as earlier versions of the program were unprotected and more convenient to use with a single disk drive.

The manual was well-written, although too concise at times for anyone unfamiliar with terminal operations. The beginner will probably require several readings of the manual to get the most out of it. The printing and binding left something to be desired, the former being small and fuzzy, the latter being flimsy. Our copy fell apart well before we were finished with it. The program is user-friendly, providing prompts when needed. Most users will rarely consult the manual after a few on-line sessions.

Microtechnic Solutions Inc., P.O. Box 2940, New Haven, CT 06515.

—Morton A. Kevelson

GREEN ARROW

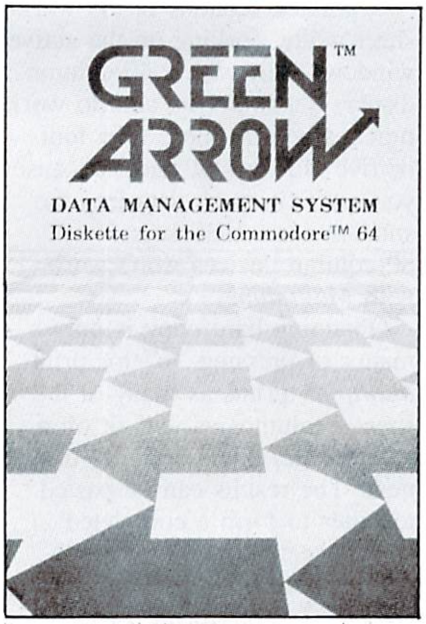
Softwave
C-64
Disk

Because computers can store and provide easy access to vast quantities of information, they have become the file cabinets of the eighties. A single 5¼" disk can store literally hundreds of pages of information and feed any of it back within seconds. Even so, it takes well-written, user-friendly software to unlock a computer's data management capabilities.

Green Arrow is a file management system for the home which comes with three record formats—Mailing List, Appointment Book and Recipe Box. Each format contains a specific list of fields that you can write to. For example, the Mailing List includes fields for company name, contact, address, phone number, etc. All fields can be easily altered to suit your needs.

In addition, each field can be

coded to accept only a limited amount of a specific type of data. This way, you can cut down on errors by specifying that the zip code will only accept numbers and there must be five of them. Also, by limiting the amount of data in a field, you can increase the total number of files a disk will hold. The computer conveniently keeps track of this number so you always know how much room you have left on a disk.



Comes with three record formats.
READER SERVICE NO. 269

Once you've created many files in a certain format, you can select which files you want to review or print in a number of different ways. First of all, you can select by any of the individual fields such as name, city, state, or zip code. Then you can limit the selection by making the contents of the selected fields less than some value, greater than some value, equal to a list of values, not equal to a list of values, or equal to a range of values. These provide almost unlimited types of selection such as "all files with addresses in New York," "all files with zip codes in a certain range," "all recipes with either pork or chicken as the main ingredient," etc.

You can also decide which field the records will be sorted by once they've been selected. Finally, you define the print format to control exactly how the records will be displayed on your monitor, or printed by your printer. Again, you're given a wide range of controllable parameters including label width and height, number of labels across, number of print lines per label, and whether or not you want to empty fields to print as blank lines. You can even include words and punctuation in the print format that don't appear in the actual files.

So you say you need a filing system that will handle more than your mailing lists, appointments, and recipes. You also want to catalogue your family tree, medical history, and collection of 11,000 baseball cards. Well, *Green Arrow* has a fourth option that allows you to design your own formats from scratch. This, along with its easy-to-use, menu-driven operating system and excellent instruction booklet, makes *Green Arrow* a fine data management system for home use.

Softwave, 156 Drakes Lane, Summertown, TN 38483 (615-964-3573).

—Lloyd Davies

MULTIPLAN
HesWare
C-64
Disk

Multiplan's name is derived from its ability to link several worksheets together. No other electronic spreadsheet includes this feature, which provides practically unlimited space for detailed small business records and financial analysis. It's equally at home with family budgets and any other application that requires a grid of numbers and text. (You can even track statistics of your favorite baseball team.)

The overall worksheet stretches 255 rows deep by 63 columns across. A three-line menu of 20 commands is always present near the bottom of the screen, and a message line below it shows error messages, prompts, and the proposed responses that guide you through the command you've selected. On the bottom of the screen, the location of the currently active cell is displayed. To the right of that, the contents of the active cell will be posted. This may be a number, text or the formula for determining the contents of that cell. At the far end of the status line, the percentage of free space remaining on the current worksheet and its name are noted.

The active cell is highlighted, and the cursor keys facilitate scrolling in all four directions. You can also page across to the next full screenful of text in any direction. Moving rapidly to a specific cell is accomplished by punching up the "GOTO" command and entering the address of the destination cell as the "proposed response." If you've used text to name certain cells something like "Sales" or "Gross Profits", you can key in the name as a response and shoot right to it. To enter text into a cell, the "Alpha" command is selected by pressing "a". The name is then keyed in, and

Upcoming *Ahoy!* Reviews
IFR Flight Simulator • Superbase 64 • Voyage of the Mayflower • C.A.R.S. • Early Games for Young Children • Sargon II • Spread Sheet Assistant • Pitstop

appears in the cell after you hit return. Numbers are entered by selecting "Value" or just typing in a number—*Multiplan* recognizes that you want to enter a number and immediately responds.

All the commands follow a similar process, offering proposed responses that request information on how to carry out the command you've punched up. Most spreadsheets force you to memorize the keystrokes for the various commands, or refer to a "cheat sheet" until you've memorized them. *Multiplan's* omnipresent menu and the step-by-step structure of its proposed responses make it an exceptionally smooth program to master and operate.

Like a word processor for numbers, *Multiplan* can insert, delete, move, and sort rows and columns. Creating a worksheet is expedited by copying the contents of a cell across, down, or to a remote location on the sheet. The formatting of a cell's contents may be varied in a number of ways, including centered, left, or right alignment. Saving and loading are as conveniently implemented as the rest of *Multiplan's* features.

Cursor movement dictates the construction of a formula. Say you want a cell to contain the total of the two cells directly above it. Cursor over to that cell, then press "=" (to tell *Multiplan* you want to enter a formula), move two cells up and press "+". Now move the cursor down one cell and press return. This formula (cell 3 equals the contents of cell 1 + the contents of cell 2) is now entered in cell three, and can be seen on the status line whenever that cell is active. When numbers are entered into the first two cells, their sum will automatically appear in the third cell. Cursor control for building formulas makes the task manageable, and all the C-64's standard mathematical operations are available. If you've defined certain cells by name, formulas may be composed of names alone, as "Profit = Sales − Costs". Few spreadsheets are sharp enough to comprehend formulas written in plain English.



Versatile spreadsheet for the C-64. READER SERVICE NO. 253

Multiplan's arsenal of 31 functions includes 12 unique ones, such as LEN and MID, in addition to SUM, AVERAGE, and others that serve as shortcuts when creating formulas. The recalc feature, which automatically recalculates every figure in the worksheet as soon as the number in any cell is changed, can be switched off to save time when altering the values of several cells. Because it's not entirely RAM-resident, *Multiplan* usually takes five to ten seconds to access the disk before completing recalculation or executing most commands.

But suppose you hit the wrong key in the middle of a formula, or select an improper command? No problem, because you can cancel *any* ongoing action by pressing the RUN/STOP key. Extensive "Help" screens are especially valuable: you can request information on specific commands, proposed responses, editing, common problems, and other topics, in addition to paging through the entire Help file.

Multiplan's most noticeable drawback is related to the C-64's 40-column display, which limits the number of columns visible at any one time. This has been somewhat circumvented by the ability to open up to eight windows, allowing you to view widely separated sections of the worksheet while working on the active window. Still, with a 40-column display you won't be able to work practically with more than four or five windows at once, because you can't fit much data into the smaller ones. HES says 80-column devices won't work with *Multiplan*.

Obtaining hard copy is only a matter of pressing "p" for print. *Multiplan* prints as many of the sheet's columns as will fit on a single page, then continues on the next. The results can be pasted together to form a completed spreadsheet that's the same dimensions as the one created onscreen. (It won't print, or even display, graphs or histograms).

The 422-page manual contains a lucid tutorial, reference section, and several useful appendices—and is nowhere as intimidating as its size makes it seem. A plastic overlay that fits over the function keys is also included, as well as a handy reference guide. Even if you never harness the full powers of what is certainly one of the most versatile spreadsheets available for the C-64, its ease-of-use and foolproof design make *Multiplan* ($99.95) an outstanding value.

HesWare, 150 North Hill Drive, Brisbane, CA 94005.

—Shay Addams

BUBBLE BURST
Spinnaker Software
C-64
Cartridge, joystick

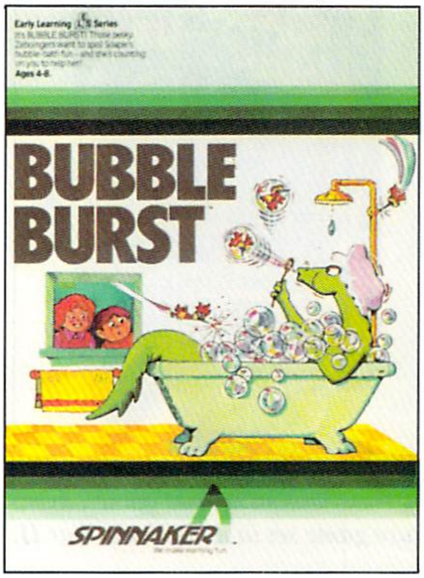
Soapie the Serpent is the main character in this "fun and learning" game for ages 4 to 8.

Soapie is discovered in repose. All she wants is to soak in her bubble bath and be left alone, but her peace is disturbed by a flock of zeboingers (birds, to you). They fly in the bathroom window and peck at Soapie's soapbubbles. If all of Soapie's bubbles are obliterated, the game is over. Soapie defends her modest domain with a bubble blower. The player lines up a zeboinger in the bubble blower's round sights, hoping to intercept the path of the pest, and pushes the fire button. If the shot is accurate, the bird is enfolded in a bubble and is carried gently out of the action.

Bubble Burst contains one- or two-player options, two skill levels, and over 25 rounds of play. The speed of the birds, and their onscreen number at any one time, increase as gameplay progresses.

It is Spinnaker's hope that the game will develop a child's ability to recognize patterns and then to predict behavior based on what they have learned. The various types of zeboingers do have distinctive flight patterns. In addition, the later, faster rounds require the child to employ some simple strategies in shot selection and in saving the limited number of bubble-salvos.

Ahoy! submitted *Bubble Burst* to the scrutiny of a fresh-faced, pure of heart six-year-old; but alas, a six-year-old somewhat jaded by exposure to computergames aimed primarily at teens and adults. Our six-year-old was diverted for a while. He had no trouble with the fundamentals of zeboinger-zapping and very much enjoyed the game for awhile, especially the way that Soapie grinned in his direction with every successful strike. However, once the concepts were fully grasped and gameplay wore on, the game grew tedious rather





Cleanliness is next to impossible with a flock of zeboingers around.
READER SERVICE NO. 237

quickly. The child showed no interest in developing a strategy, even under encouragement, except in regards to the zeboinger whose strategy is to turn on the shower. Increasing the difficulty didn't seem to help. The child left the game with little reluctance and when a second session was offered some time later, the response was tepid.

Bubble Burst might be fine for younger children or children not normally inclined to play arcade-style games. At the same time, gameplay *is* a one-note affair at any speed or difficulty.

The game booklet contains a cleverly illustrated and readable history of Soapie The Serpent as well as a tutorial on bubbles, how to make bubble blowers, how to create odd-shaped bubbles, and so on. The documentation is excellent, but don't expect a long life for the game itself.

Spinnaker Software, 215 First Street, Cambridge, MA 02142.

—Tim Moriarty

CASTLE WOLFENSTEIN
Muse Software
C-64
Disk; joystick, keyboard

Achtung! Was ist das? Feuer!

Huh? Wazzat? Only the TV? I thought Mr. Roboto was sneaking up on me!

My Commodore talks—threatens, really. Its verbosity (all of ten words) is the fault of a new/old game named *Castle Wolfenstein*. New for the C-64. Old because the Apple version has been out for two and one-half years, the Atari version for two.

It is mid-WWII. *Castle Wolfenstein* has been converted by the Nazis into a prison/supply dump/headquarters. You are an allied POW (and this ain't *Hogan's Heroes*—you *want* to escape!). Deep in the bowels of the castle, a fellow prisoner bequeaths you his advice and his only valuable possessions before the SS take him away—for good. His advice? You must escape. One prisoner almost made it when he found a Nazi uniform. But first, find the German war plans. You'll save thousands of lives. His possessions? A handgun and ten bullets. His death sceam still ringing in your

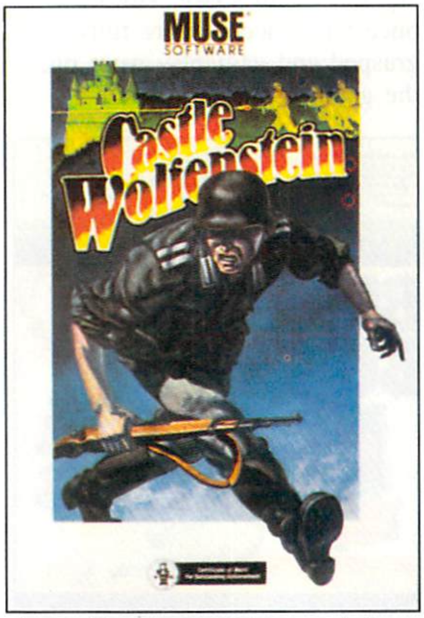
ears, you begin your life-or-death adventure.

Seeing only one room of the Castle onscreen at a time, you must trek through damp corridors filled with locked doors, Nazi guards, SS storm troopers, and supply chests. The guards are easy. They walk routine patrols. You can shoot them from around a corner or they will surrender if you confront them at point-blank range. Search guards (dead or alive) to get their keys and extra bullets. It is ironic to force a guard into surrendering when you have no bullets. Once you have taken his, shoot him. You really have no choice; you almost have to leave a trail of dead bodies strewn behind in order to beat this game.

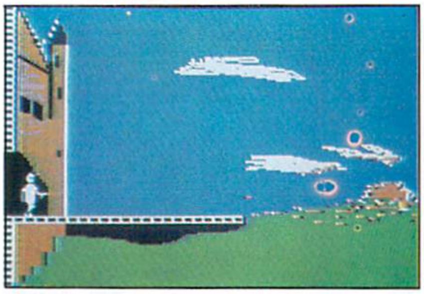
The SS storm troopers (those aren't letter sweaters they're wearing) are much more formidable foes. Bullet-proof vests are their standard issue, so your handgun will not be much use. You will have to use a grenade—if you can find one. Of course, if you are too close when the grenade goes off. . . But you don't mind starting over. The SS patrol no posts; rather, they roam the castle with apparently nothing to do but blow away escaped prisoners. Once you get the SS on your tail, one of you will die before the chase ends.

Each room of the castle will contain a chest or two. Supply chests are the most essential element of this game—and the most aggravating. To open a chest, stand next to it, point at the lock, and wait. Maybe one minute. Maybe four. Once the seconds tick away, the chest will pop open revealing grenades or bullets, or even a German uniform or bullet-proof vest, if you are lucky. Of course, that long wait may culminate in nothing more than schnapps or bratwurst, so do not expect too much.

You can speed up the process of opening chests by shooting them. (You also can end the game by shooting a chest full of explosives.) Shooting at locked doors may also open them, but first try the keys you have taken from dead Nazis. You will need your bullets later and sometimes even ten will not open a door.



A combination arcade game/adventure game set in mid-World War II.



Elude the SS and escape.
READER SERVICE NO. 246

Castle Wolfenstein presents you with several ways to die—just like real life. One of these ways is far superior to the others—unlike real life. If you are caught and shot by Nazi guards, the game will start over. You will be in the first room, but your progress will be saved. Open chests will still be open and dead guards will still be dead. Other ways of dying, like in a grenade explosion, force you to really start over. Even the castle's floor plan will be different.

Castle Wolfenstein has few faults—none if you are patient. Each room must be loaded from disk so there is a lag as you move through the castle. Every time you bump into a wall, the screen goes into hysterics for a few seconds. After about two bumps, you will wish there were a way to turn off this "feature." Then, there are the chests. Once they are opened they stay open, but sometimes you'll decide to go for a sandwich while you are waiting. You can play from the keyboard or with a joystick. Use the joystick! You will still have to hit a few keys to fire your pistol, throw a grenade, etc.

This game, despite its periodic lethargy, is addicting. I am not all that big on killing everything that moves, but I really get caught up in the adventure—drawing a map, trying to outflank guards, hoping this chest will not contain more bratwurst—as the game says, "AWFUL." *Castle Wolfenstein* was one of the first combination arcade/adventure games. Using simple but effective graphics, it tests your arcade reflexes and your problem-solving skills.

Muse Software, 347 N. Charles Street, Baltimore, MD 21201.

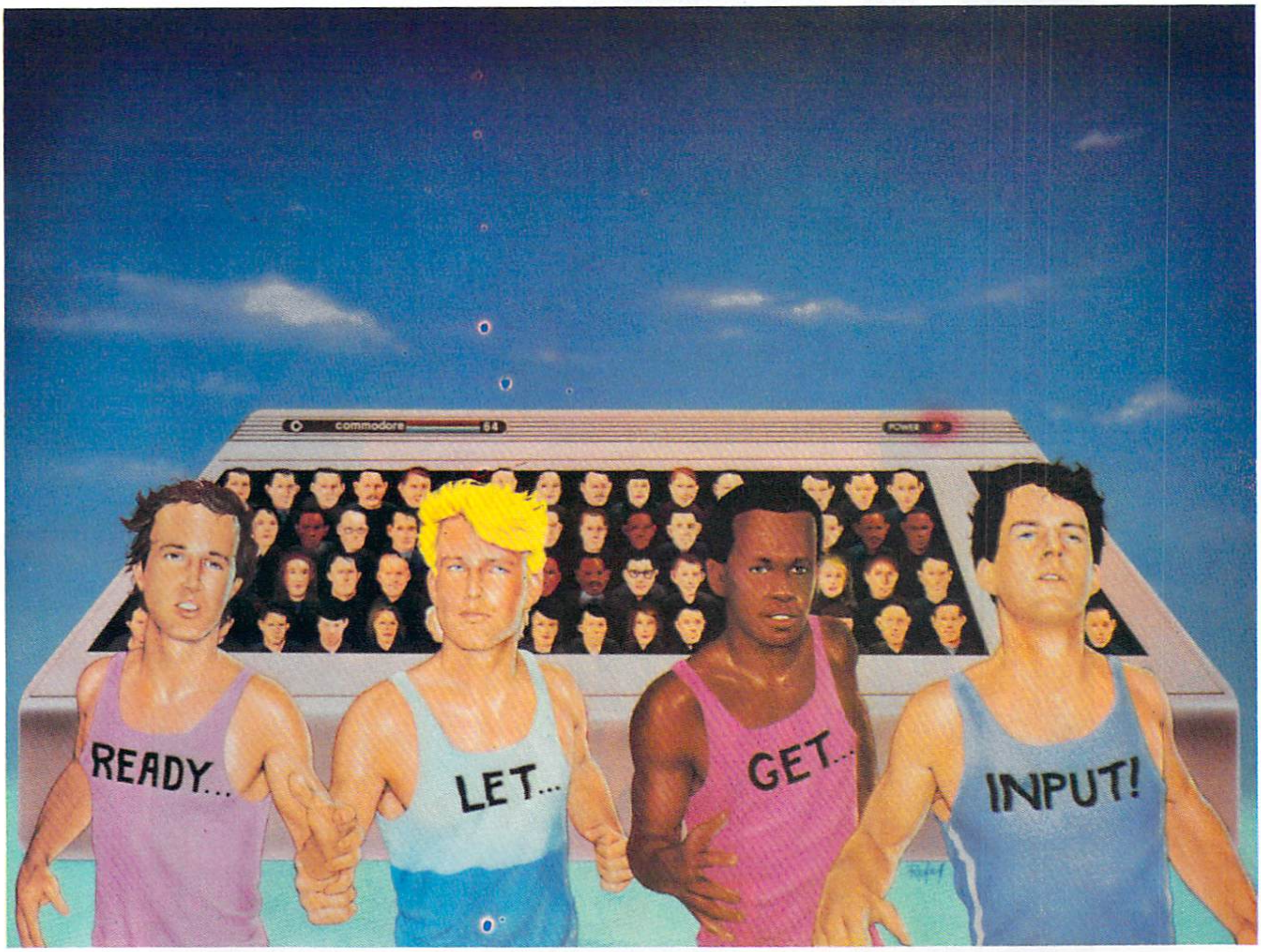
—Richard Herring

SIMULATED COMPUTER II

Carousel Software, Inc.
C-64
Disk, cassette; keyboard

Everybody who buys a home computer takes a stab at learning to program. Most of us started

Continued on page 87



© James Regan 1984

AN ARTICLE ON INPUTTING DATA

By Dale Rupert

Quickly—how many ways can you think of to put data into your computer program?

That should be easy. Let's see. There is the seldom-seen LET statement: LET A = 35. That gives a value of 35 to the variable A. (Of course, the shorter form A = 35 does just as well.)

Then there is the INPUT statement. It allows the person running the program to enter data into the program from the keyboard:

```
10 INPUT "WHAT IS YOUR NAME" ; N$
```

When the computer executes this line, it prints whatever is within the quotation marks followed by a question mark. Then the program stops and waits for the user to type something and to press the Return key. Subsequently that something is assigned to the string variable N$.

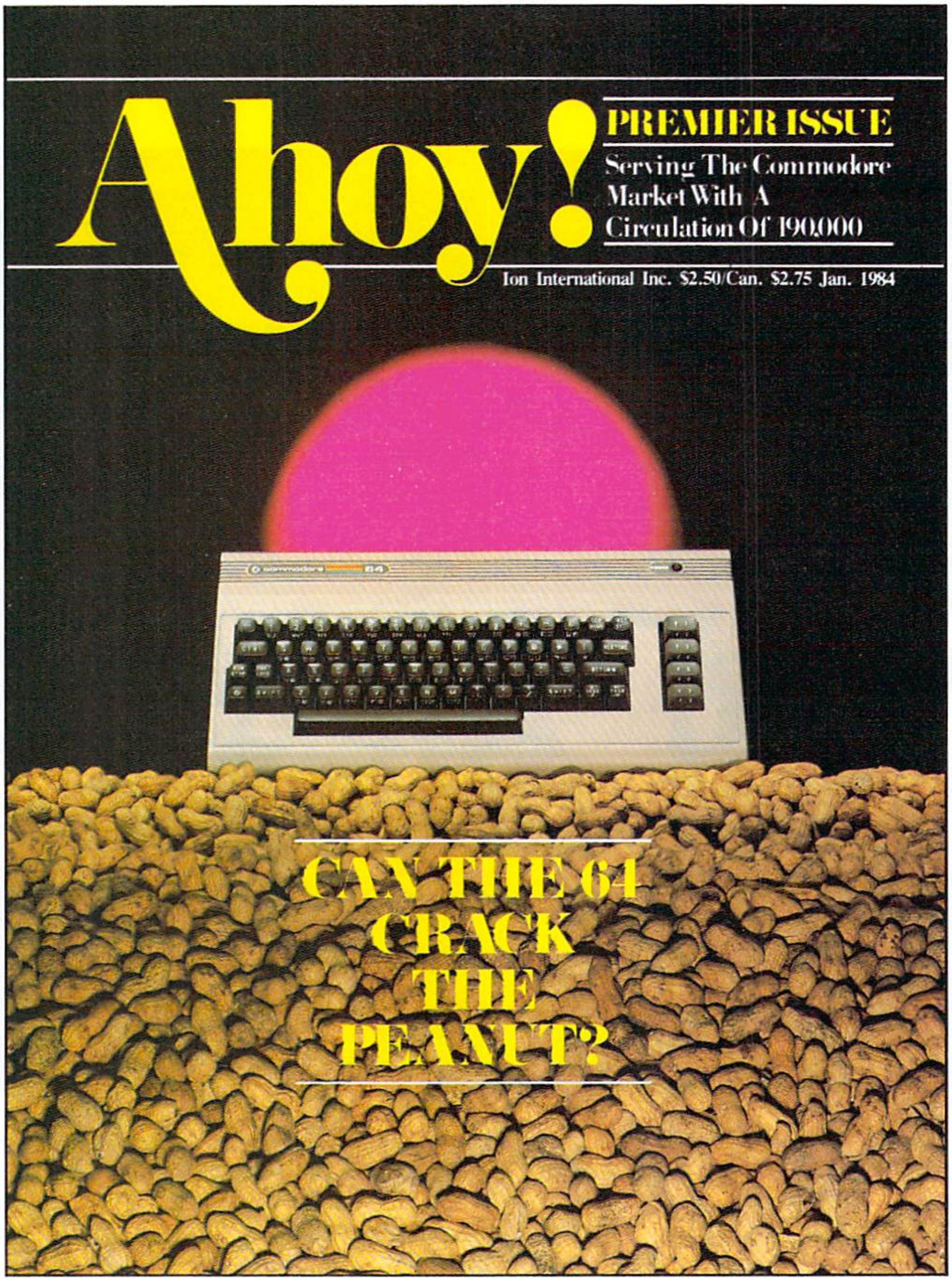
Another common way to "give" data to a program is with the READ and DATA statements. Just as with the LET statement, the programmer stores the data in the program when it is written. Contrast that with the INPUT statement which brings data into the program when it is run.

The READ/DATA statements work like this:

```
20 READ A,B$,C
30 DATA 23,HELLO EVERYONE,17
```

Here the first item in the DATA list (23) is assigned to the numeric variable A. The second item is the string HELLO EVERYONE, and B$ is given that value. The final value of 17 is assigned to C.

Notice a few things about READ/DATA. The items in the DATA list must correspond with the



HAS EVERYTHING!

SUBSCRIBE TO AHOY!

☐ Twelve Issues for $19.95 ($26.95 Canada and elsewhere)
☐ Twenty-four Issues for $37.95 ($49.95 Canada and elsewhere)

Name______________________________

Address______________________________

City______________ State______________ Zip______________

Send coupon or facsimile to:
ION INTERNATIONAL INC.
45 West 34th Street, Room 407, New York, NY 10001

variable types of the items in the READ statement. Replacing the B$ with a B results in a ?SYNTAX ERROR IN 30 response. Also, it is not necessary to enclose the string value within quotes (unless it contains a comma), but quotation marks are permitted.

Finally, the items in the DATA statement are separated from one another by commas. There may be more DATA items than READ items, but if there are fewer DATA items, you will see the infamous ?OUT OF DATA ERROR IN 20 reply. The data items need not all be contained in the same DATA statement. The computer begins at the top of your program looking for DATA statements when it encounters a READ statement. It continues searching for DATA statements throughout your program until it has found a value for all variables in the READ statement.

While we are dealing with READ/DATA, we may as well mention the associated RESTORE statement. RESTORE allows the items in a DATA statement to be read more than once. It merely causes the computer to start at the top of the program once again to locate the first DATA statement and value it can find.

What values do A, B and C have after this line is executed?

```
5 READ A,B:RESTORE:READ C:DATA 5,
6,7
```

If you said "5, 6, 7", you should review the RESTORE statement. If you said "5, 6, 5", then you are ready for the "no-holds-barred, try to confuse 'em data input mastery" quiz. Without using your computer, predict what the following program will print:

```
10 REM - LET, INPUT, AND READ/DAT
A QUIZ
20 REM <THE USER TYPES 15 OAK IN
RESPONSE TO THE FOLLOWING QUESTIO
N>
30 INPUT "WHAT IS YOUR ADDRESS";A
$
40 READ A,B,C
50 READ D
60 RESTORE : READ E
70 READ F$
80 LET G=VAL(A$)
90 LET H=VAL(F$ +F$)
100 PRINT A,B,C,D,E,F,G,H
110 DATA 5,10
120 DATA 15,20,25
```

Later in the article I'll list the answers. If you must, go ahead and type it into your computer. If you don't get the right answers, then you *should* run it on your computer.

The three types of data input statements just discussed are the most commonly used ones. Each has its own peculiarities and advantages.

There are still at least four other statements that can be used to bring data into a computer program. How many can you think of? Probably the next most common is the GET statement.

The GET statement is usually used to input a single alphanumeric (string) character into the program. GET reads one character from the keyboard buffer and assigns that value to a variable: GET A$ removes the first item from the buffer and stores it in A$.

The keyboard buffer is a set of ten RAM locations starting at address 631 in the Commodore 64. As you type, the value of each key is stored in the buffer temporarily. The computer periodically reads this buffer to see if anything is waiting there. That's what happens unless the computer is busy doing something else. Run this little program to keep the computer busy:

```
5 FOR P=1 TO 5000 : NEXT
```

While it is running, rapidly type a series of letters on the keyboard. They won't appear on the screen until after the computer is done running our program. When the program is done, you will see at least part of your letters below the READY prompt. If you count them, you will see that only the first ten letters appear. The later ones fell into the proverbial "bit-bucket" and are lost.

The advantages of a buffer should be apparent. Even if you type faster than the computer can read, the computer can eventually catch up with you. If you type LIST [RETURN] while the program above is running, the computer will obey your command once it finally has a chance to read it.

The computer executes a GET A$ instruction by removing the next character from the buffer if one is there at that moment. If so, the character is assigned to A$. If the buffer happens to be empty when the GET statement is executed, A$ is assigned a null value.

The null value is represented the same way as other string constants, by the use of quotation marks. But the null value (or null string) simply has nothing between the quotation marks: "". This is not to be confused with the space character " " which merely looks as though it has nothing between the quotes. The space character takes up room on the screen whereas the null string does not.

If you take the ASCII value of the space character, you get 32. But all you will get is an "?ILLEGAL QUANTITY ERROR" if you type PRINT ASC("") using the null string. What do you suppose LEN("") gives? Try it and see.

Although both bring data into a program from the

keyboard, there is a primary difference between the GET and the INPUT statements. INPUT puts the computer into a temporary holding pattern. The computer continuously scans the keyboard buffer, pulling characters out and printing them as the user types them in. Execution continues with the next program statement only after the computer reads a (return) code, CHR$(13), from the buffer. Consequently the user tells the computer when the INPUT data is complete by pressing the [RETURN] key.

On the other hand, the GET statement is a "one time only" command. If there happens to be nothing in the buffer when the computer executes GET A$, then that is exactly the value assigned to A$—the null string. Consequently GET is frequently used within is loop such as in this program:

```
10 REM - USING THE GET STATEMENT
20 PRINT "PLEASE TYPE 5 CHARACTER
S"
30 GET A$ : IF A$ < > "" THEN 30
40 GET B$ : IF B$ = "" THEN 40
50 PRINT B$;
60 C$ = C$ + B$ : R$ = B$ + R$
70 IF LEN(C$) < 5 THEN 40
80 PRINT :PRINT R$ " IS THE REVER
SE OF " C$
```

This program shows that the programmer has more control over the input data with a GET statement. This program accepts only the first five characters entered from the keyboard. Also, the [RETURN] key is not necessary here.

Notice that the logical tests in lines 30 and 40 are opposites. Line 30 reads the buffer until it is empty, thus clearing out any characters that may be there. This is a very good idea if you are using the GET statement in the middle of a program. As we saw earlier, the buffer may have accumulated unwanted keystrokes while the computer was busy doing something else.

Now that the buffer is empty, line 40 sits and waits for a keystroke to occur. B$ will no longer equal the null string once a key is pressed, and execution then continues with line 50.

Line 50 prints (or echoes) the typed character. Notice that unlike INPUT, GET does not print the characters that are typed from the keyboard. GET does not even show the cursor. Those niceties are up to the discretion and expertise of you, the programmer.

Line 60 builds up a string variable C$ by concatenating the characters. R$ is also built up as the keys are pressed, but each new character is put onto the front of R$. As a result, C$ and R$ are opposites of one another. The mathematicians among you will conclude that concatenation is not commutative. Everyone else will realize that the order in which strings are joined makes a difference.

Line 70 is the watchdog to make sure the string doesn't get too long, and line 80 prints the results.

Why would you use GET if INPUT does so much automatically? Once again, GET allows the programmer to better control the characters that are typed into the program. Improper characters can easily be handled or ignored, as this example shows:

```
10 PRINT "TYPE ANYTHING EXCEPT 'Q
'"
20 GET A$ : IF A$="" THEN 20
30 IF A$ = "Q" THEN PRINT : GOTO
10
40 PRINT A$ : GOTO 20
```

Line 20 causes the computer to sit in a loop until a key is pressed. If that key is a "Q", then line 30 puts the cursor onto the next screen line with the PRINT statement and branches back to line 10 where the initial message is reprinted. If the typed character is not a "Q", then the IF statement in line 30 is false. The rest of line 30 is ignored, and execution continues at line 40. Here the input character is echoed on the screen, and the program goes back to line 20 to get another character.

So far all of the data input statements we have discussed have been related to the keyboard. Let's have a look at a slightly more exotic way of bringing data into a program.

Let's investigate the PEEK function. PEEKing allows us to get data directly from memory. Descriptions of many memory locations are listed in the *Commodore 64 Programmer's Reference Guide* beginning on page 310. Here is a program that can look at itself in memory by means of the PEEK function:

```
10 REM...BASIC MIRROR
20 M1 = PEEK(43)
30 M2 = PEEK(44)
35 REM << M = START OF TEXT
40 M = M1 + 256 * M2
45 REM << N = START OF VARIABLES
50 N = PEEK(45) + 256 * PEEK(46)
60 FOR P = M TO N-1
70 V = PEEK(P)
80 IF V < 31 OR V > 128 THEN V =
166
90 PRINT P , PEEK(P) , CHR$(V)
100 NEXT P
```

A pointer to the starting address of BASIC text (namely this program) is stored as two bytes in memory locations 43 and 44 according to the last line of page 311 in the *Programmer's Reference Guide*. The value of a two-byte pointer is calculated

Continued on page 95



COMMODARES

PROGRAMMING CHALLENGES

By Dale Rupert

Each month, we'll present several programs designed to toggle the bits in your cerebral random access memory. We invite you to send your solutions to *Commodares,* P.O. Box 723, Bethel, CT 06801. We'll print and discuss the cleverest, simplest, shortest, or most unusual solutions that we receive. Your original programming problems would be equally welcome!

PROBLEM #6-1 : SPEED DEMON

Write the fastest program to assign the values 1 through 10 to the variables A through J respectively.

```
1 TA=TI
2 FOR NN=1 TO 100
10 .
```

(your program goes here)

```
900 .
1000 NEXT
1010 TB=TI
1020 PRINT TB-TA "JIFFIES FOR 100
 ITERATIONS"
```

Your program must start at line 10 in the program above. Lines 1, 2, and 1000-1020 must be typed just as shown. This timing routine causes your program to be executed 100 times to give a more accurate measurement of your program's speed.

PROBLEM #6-2: SIMPLE MYSTERY

Without typing it in, figure out what the following short program does:

```
10 GET A$ : IF A$="" THEN 10
20 A=ASC(A$)
30 IF A=65 OR A=69 OR A=73 OR A=7
9 OR A=85 THEN PRINT"{RV}" A$ "{R
O}"; : GOTO 10
40 PRINT A$; : GOTO 10
```

PROBLEM #6-3: STRING CHALLENGE

For a real challenge, write a subroutine which simulates an INSTRING function. This function locates the first occurrence of the string A$ within the string B$ at which A$ starts. If A$ is not within B$, the value of N is zero.

For example, if A$="CAT" and B$="CONCATENATE", your subroutine assigns the value 4 to N since "CAT" starts at position 4 in "CONCATENATE". If A$="C" then N=1, and if A$="COT" or A$="Z" then N=0, using the same value for B$.

PROBLEM #6-4: GRAPHIC RECTANGLE

Use graphic symbols to draw a rectangle on the screen of width W and height H. W may range from 3 to 40 and H may range from 3 to 24. The boundary of the rectangle should be a single continuous line. The rectangle should be centered on the screen. The user may specify H and W.

If you were quick and had an understanding of arrays, you probably determined the answer to *Problem #5-1: Faster Computer* from last month's *Ahoy!* faster than your computer. Even using paper and pencil to get started, you could have found that the answer was 999,000 before you could have typed and run the program for the result. Sometimes the built-in computer is better.

Problem #5-2: Random Array created then scanned a random array of numbers looking for the index or subscript of the largest A(L) and smallest A(S) elements of the array. Often it is handier to work with just the subscripts rather than the actual values of the elements in an array.

One solution to *Problem #5-3: Running Average* is listed below.

```
5 N=0
10 INPUT V
20 S=S+V
30 N=N+1
40 PRINT "WITH" N "TERMS, THE AVE
```

```
RAGE IS" S/N
50 GOTO 10
```

If you add line 15,

```
15 IF V<0 THEN N=N-2
```

then you may delete a value by typing its negative. This assumes that all other values you are using are positive.

You can solve *Problem #5-4: Letter Triangle* with the following program:

```
10 FOR L=1 TO 26
20 FOR N=1 TO L
30 PRINT CHR$(L+64);
40 NEXT N
50 PRINT
60 NEXT L
```

Because of the time lag between deadlines and publication dates, we are just now receiving responses to the *Commodares* in the March issue of *Ahoy!*

The shortest solutions to *Problem #2: Reversals* were one-liners of the form submitted by Dino Russo (Columbia, SC):

```
1 INPUT"WORD OR PHRASE";W$ :FOR L
=LEN(W$) TO 1 STEP -1 :PRINT MID$
(W$,L,1); :NEXT
```

Tim Forget (Kirkland Lake, Ontario), Gregg Leonard (Dayville, CT), Robin Robertson (Alhambra, CA), Jason Franzen (Liberal, KS), Lionel Sapkus (Burbank, IL), Lloyd Burkett (Laurelton, NY), Bob Fultz (Clarksburg, WV), and Mark Nekic (Eastlake, OH) all submitted similar programs.

S. Gaudet (Rumford, ME) very cleverly added TAB(20-LEN(N$))/2 between the PRINT and the MID$ in the program above. Can you figure out what that does? It's a nice touch if N$ isn't too long. Can you modify this function to center the results on the C-64's forty column screen?

Ken Buskirk (Lancaster, CA) submitted a program for REVERSALS which is paraphrased below:

```
10 X=39 : B$=CHR$(32) :C$=CHR$(13
)
20 GET A$ : IF A$="" THEN 20
30 X=X-1 : PRINT SPC(X) A$ "{CU}"
40 IF A$=B$ AND X<=10 THEN X=39:P
RINT
50 IF X=0 THEN X=39:PRINT
60 IF A$=C$ THEN X=39 : PRINT"{CU
}"
70 GOTO 20
```

Ken puts the letters on the screen from right to left as you type them. RETURN starts a new line as does the space bar if you have fewer than 10 spaces left on a line. It's quite an experience to run this program. Give it a try. Ken said it's one of his first programs. Keep up the good work!

Several readers sent programs which included *Reversals* and *Palindromes* in one. They include Barbara Steinman (New York, NY), Joe Persichetti (Trenton, NJ), Larry Masterson (Willard, OH), Robert Griffiths (Syracuse, NY), and James Dunavant (Gainesville, FL). Mr. Dunavant pointed out that our zip code is a palindrome!

Of all the *Palindrome* solutions, Mark Nekic's program was the shortest:

```
1 INPUT N$:L=LEN(N$):FOR X=1 TO L
:IF MID$(N$,X,1)=MID$(N$,L+1-X,1)
 THEN NEXT:PRINT "YES"
```

(The spaces have been added for clarity.)

The two logic puzzles *Logical Conclusion* and *Illogical Conclusion* brought in far fewer solutions than the other two problems. Congratulations to Dino Russo, Mark Nekic, and Daniel Amodeo (Bethesda, MD) for their work on these problems.

Thanks to everyone else who submitted programs and comments. Keep up the good programming! (And please note the new address for *Commodares,* listed at the beginning of page 63.) □



"Can it remember anniversaries?"

PROGRAM LISTINGS

On the following pages are listed several programs that we hope you'll want to punch in your Commodore computer. But please read the following introduction first; there are a few things you'll need to know.

Certain computer commands are displayed on the monitor by a variety of odd-looking characters. To get your computer to display these commands rather than actually perform them, you'll need to enter the quote mode. Hold down the SHIFT key and press the "2" key; a set of quote marks will appear. This tells the computer that the characters that follow are to be displayed, not performed. To exit the quote mode, type another set of quote marks, or hit the RETURN key. You'll also enter the quote mode when you INserT spaces or characters onto a line.

In *Ahoy!'s* program listings, you'll frequently find letters and/or numbers surrounded by brackets { }. That's because, for the purposes of clear reproduction, we at *Ahoy!* use a daisy wheel printer incapable of reproducing command symbols. For example, when you're in the quote mode and press the SHIFT and CLR/HOME keys at the same time, the screen (or a dot-matrix printer) will indicate this command with a heart {♥}. Because a daisy wheel cannot duplicate this symbol, it substitutes an alternate code between brackets. In the case of the SHIFT/CLR HOME symbol, our printer substitutes {SC}.

Another special case is SHIFT and COMMODORE characters. We represent these by underlining or overlining, respectively: any character underlined in the program listing should be punched in as a SHIFTed character ($\underline{J}$ = SHIFT J), any character overlined should be punched in as a COMMODORE character ($\bar{J}$ = COMMODORE J).

An alternate way of entering commands and other graphics symbols and characters is to use their corresponding character strings. The CLR/HOME command, for example, is entered by typing CHR$(147). While this requires a few extra strokes, it facilitates editing your program or reading the printed listing. For a complete list of CHR$ codes, consult the appendix at the back of your Commodore user manual.

Below is a list of the command abbreviations you'll find in our program listings, the commands they stand for, how to enter them, and how they'll appear on the screen or on a dot matrix printout.

When You See	It Means	You Type	You Will See	When You See	It Means	You Type	You Will See
{SC}	Screen Clear	SHIFT CLR/HOME		{YL}	Yellow	CNTRL 8	
{HM}	Home	CLR/HOME		{OR}	Orange	COMMODORE 1	
{CU}	Cursor Up	SHIFT ↑ CRSR ↓		{BR}	Brown	COMMODORE 2	
{CD}	Cursor Down	↑ CRSR ↓		{LR}	Light Red	COMMODORE 3	
{CL}	Cursor Left	SHIFT ← CRSR →		{G1}	Grey 1	COMMODORE 4	
{CR}	Cursor Right	← CRSR →		{G2}	Grey 2	COMMODORE 5	
{SS}	Shifted Space	SHIFT space		{LG}	Light Green	COMMODORE 6	
{IN}	Insert	INST		{LB}	Light Blue	COMMODORE 7	
{RV}	Reverse On	CNTRL 9		{G3}	Grey 3	COMMODORE 8	
{RO}	Reverse Off	CNTRL 0		{F1}	Function 1	F 1	
{BK}	Black	CNTRL 1		{F2}	Function 2	F 2	
{WH}	White	CNTRL 2		{F3}	Function 3	F 3	
{RD}	Red	CNTRL 3		{F4}	Function 4	F 4	
{CY}	Cyan	CNTRL 4		{F5}	Function 5	F 5	
{PU}	Purple	CNTRL 5		{F6}	Function 6	F 6	
{GN}	Green	CNTRL 6		{F7}	Function 7	F 7	
{BL}	Blue	CNTRL 7		{F8}	Function 8	F 8	

BUG REPELLENT CORRECTION

These are updated versions of *Bug Repellent*. See the note on page 4 before typing any programs!

VIC 20 BUG REPELLENT

By Michael Kleinert and David Barron

The program listed below will allow you to quickly debug any *Ahoy!* program you type in on your VIC 20. Follow directions for cassette or disk.

For cassette: type in and save the *Bug Repellent* program, then type RUN 63000[RETURN]SYS 828[RETURN]. If you typed the program properly, it will generate a set of two-letter line codes that will match those listed below the program on this page. (If you didn't type the program properly, of course, no line codes will be generated. You'll have to debug the *Bug Repellent* itself the hard way.)

Once you've got a working *Bug Repellent*, type in the program you wish to check. Save it and type the RUN and SYS commands listed above once again, then compare the line codes generated to those listed in the magazine. If you spot a discrepancy, a typing error exists in that line. Important: you must use exactly the same spacing as the program in the magazine. Due to memory limitations on the VIC, the VIC *Bug Repellent* will register an error if your spacing varies from what's printed.

You may type SYS 828 as many times as you wish, but if you use the cassette for anything, type RUN 63000 to restore the *Repellent*.

When your program has been disinfected you may delete all lines from 63000 on. (Be sure the program you type doesn't include lines above 63000!)

For disk: type in the *Bug Repellent*, save it, and type RUN:NEW[RETURN]. (See above regarding testing the *Bug Repellent* on itself.) Type in the program you wish to check, then SYS 828. This will generate a set of two-letter line codes that you should compare to those listed in the magazine.

To pause the line codes listing, press SHIFT. To permanently pause it, press SHIFT LOCK. To continue, release SHIFT LOCK.

To send the list to the printer type OPEN 4,4:CMD 4:SYS 828[RETURN]. When the cursor comes back, type PRINT#4:CLOSE 4[RETURN].

```
•63000 FOR X = 828 TO 1023 :READ Y
:POKE X,Y:NEXT:END
•63001 DATA 169, 0, 133, 63, 133,
64, 165, 43, 133, 251
•63002 DATA 165, 44, 133, 252, 160
, 0, 132, 254, 32, 228
•63003 DATA 3, 234, 177, 251, 208,
 3, 76, 208, 3, 230
•63004 DATA 251, 208, 2, 230, 252,
 169, 244, 160, 3, 32
•63005 DATA 30, 203, 160, 0, 177,
251, 170, 230, 251, 208
•63006 DATA 2, 230, 252, 177, 251,
 32, 205, 221, 169, 58
•63007 DATA 32, 210, 255, 169, 0,
133, 253, 230, 254, 32
•63008 DATA 228, 3, 234, 165, 253,
 160, 0, 170, 177, 251
•63009 DATA 201, 32, 240, 6, 138,
113, 251, 69, 254, 170
•63010 DATA 138, 133, 253, 177, 25
1, 208, 226, 165, 253, 41
•63011 DATA 240, 74, 74, 74, 74, 2
4, 105, 65, 32, 210
•63012 DATA 255, 165, 253, 41, 15,
 24, 105, 65, 32, 210
•63013 DATA 255, 169, 13, 32, 210,
 255, 173, 141, 2, 41
•63014 DATA 1, 208, 249, 230, 63,
208, 2, 230, 64, 230
•63015 DATA 251, 208, 2, 230, 252,
 76, 74, 3, 169, 236
•63016 DATA 160, 3, 32, 30, 203, 1
66, 63, 165, 64, 32
•63017 DATA 205, 221, 169, 13, 32,
 210, 255, 96, 230, 251
•63018 DATA 208, 2, 230, 252, 96,
0, 76, 73, 78, 69
•63019 DATA 83, 58, 32, 0, 76, 73,
 78, 69, 32, 35
•63020 DATA 32, 0, 0, 0, 0, 0
```

BUG REPELLENT LINE CODES FOR VIC 20 BUG REPELLENT

```
LINE # 63000:MH     LINE # 63011:NN
LINE # 63001:BD     LINE # 63012:IG
LINE # 63002:FO     LINE # 63013:EN
LINE # 63003:ND     LINE # 63014:GJ
LINE # 63004:DJ     LINE # 63015:IK
LINE # 63005:LP     LINE # 63016:HG
LINE # 63006:JB     LINE # 63017:CK
LINE # 63007:JF     LINE # 63018:JF
LINE # 63008:KA     LINE # 63019:OH
LINE # 63009:HP     LINE # 63020:LH
LINE # 63010:KJ     LINES: 21
```

C-64 BUG REPELLENT

By Michael Kleinert and David Barron

The program listed below will allow you to quickly debug any *Ahoy!* program you type in on your C-64.

Type in and save the *Bug Repellent* program. Type NEW, then type in the *Ahoy!* program you wish to check. When that's done, save your program (don't run it!) and type SYS 49152 RETURN. You'll be asked if you want the line value codes displayed on the screen or dumped to the printer. If you select screen, it will appear there.

The table will move quickly, too quickly for most mortals to follow. To pause the listing depress and hold the SHIFT key. To pause for an extended period, depress SHIFT LOCK. As long as it is locked, the display will remain frozen.

Compare the table your machine generates to the table in *Ahoy!* that follows the program you're entering. If you spot a difference, an error exists in that line. Jot down the numbers of lines where contradictions occur, LIST each line, spot the errors, and correct them.

```
•5000 FOR X = 49152 TO 49488 :READ
 Y:POKE X,Y:NEXT:END
•5001 DATA 32, 161, 192, 165, 43,
133, 251, 165, 44, 133
•5002 DATA 252, 160, 0, 132, 254,
32, 37, 193, 234, 177
•5003 DATA 251, 208, 3, 76, 138, 1
92, 230, 251, 208, 2
•5004 DATA 230, 252, 76, 43, 192,
76, 73, 78, 69, 32
•5005 DATA 35, 32, 0, 169, 35, 160
, 192, 32, 30, 171
•5006 DATA 160, 0, 177, 251, 170,
230, 251, 208, 2, 230
•5007 DATA 252, 177, 251, 32, 205,
 189, 169, 58, 32, 210
•5008 DATA 255, 169, 0, 133, 253,
230, 254, 32, 37, 193
•5009 DATA 234, 165, 253, 160, 0,
76, 13, 193, 133, 253
•5010 DATA 177, 251, 208, 237, 165
, 253, 41, 240, 74, 74
•5011 DATA 74, 74, 24, 105, 65, 32
, 210, 255, 165, 253
•5012 DATA 41, 15, 24, 105, 65, 32
, 210, 255, 169, 13
•5013 DATA 32, 220, 192, 230, 63,
208, 2, 230, 64, 230
•5014 DATA 251, 208, 2, 230, 252,
76, 11, 192, 169, 153
•5015 DATA 160, 192, 32, 30, 171,
166, 63, 165, 64, 76
•5016 DATA 231, 192, 96, 76, 73, 7
8, 69, 83, 58, 32
•5017 DATA 0, 169, 247, 160, 192,
32, 30, 171, 169, 3
•5018 DATA 133, 254, 32, 228, 255,
 201, 83, 240, 6, 201
•5019 DATA 80, 208, 245, 230, 254,
 32, 210, 255, 169, 4
•5020 DATA 166, 254, 160, 255, 32,
 186, 255, 169, 0, 133
•5021 DATA 63, 133, 64, 133, 2, 32
, 189, 255, 32, 192
•5022 DATA 255, 166, 254, 32, 201,
 255, 76, 73, 193, 96
•5023 DATA 32, 210, 255, 173, 141,
 2, 41, 1, 208, 249
•5024 DATA 96, 32, 205, 189, 169,
13, 32, 210, 255, 32
•5025 DATA 204, 255, 169, 4, 76, 1
95, 255, 147, 83, 67
•5026 DATA 82, 69, 69, 78, 32, 79,
 82, 32, 80, 82
•5027 DATA 73, 78, 84, 69, 82, 32,
63, 32, 0, 76
•5028 DATA 44, 193, 234, 177, 251,
 201, 32, 240, 6, 138
•5029 DATA 113, 251, 69, 254, 170,
 138, 76, 88, 192, 0
•5030 DATA 0, 0, 0, 230, 251, 208,
 2, 230, 252, 96
•5031 DATA 170, 177, 251, 201, 34,
 208, 6, 165, 2, 73
•5032 DATA 255, 133, 2, 165, 2, 20
8, 218, 177, 251, 201
•5033 DATA 32, 208, 212, 198, 254,
 76, 29, 193, 0, 169
•5034 DATA 13, 76, 210, 255, 0, 0,
 0
```

BUG REPELLENT LINE CODES FOR C-64 BUG REPELLENT

```
LINE # 5000:GJ      LINE # 5018:FK
LINE # 5001:DL      LINE # 5019:FL
LINE # 5002:DB      LINE # 5020:CL
LINE # 5003:OF      LINE # 5021:GC
LINE # 5004:KN      LINE # 5022:NN
LINE # 5005:CA      LINE # 5023:NH
LINE # 5006:CE      LINE # 5024:IM
LINE # 5007:JE      LINE # 5025:KC
LINE # 5008:CL      LINE # 5026:DC
LINE # 5009:NB      LINE # 5027:ML
LINE # 5010:MB      LINE # 5028:GN
LINE # 5011:EP      LINE # 5029:JK
LINE # 5012:GH      LINE # 5030:NA
LINE # 5013:AN      LINE # 5031:DM
LINE # 5014:NG      LINE # 5032:JA
LINE # 5015:BF      LINE # 5033:FM
LINE # 5016:EP      LINE # 5034:PA
LINE # 5017:PJ      LINES: 35
```

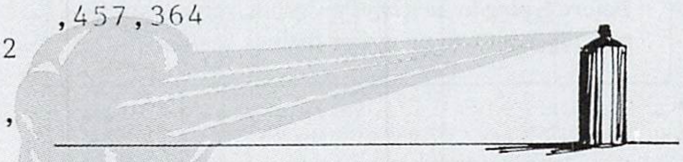
SOUND CONCEPT MAKER

FROM PAGE 98

> **IMPORTANT**
> **Before typing in an *Ahoy!* program, refer to the first two pages of the program listings section.**

```
•10 POKE36879,31:PRINT"{SC}{CD}{CD
}{CR}{BL}{RV}SOUND{RO} {RV}CONCEP
T{RO} {RV}MAKER{CD}"
•20 PRINT"{BK}{CD}{CR}{RV}OPTIONS:
"
•30 PRINT"{CD}{CR}{RV}1.{RO}  PLAC
E {RV}SOUNDER{RO}{RO} IN":PRINT"{
CR}{CR}{CR}{CR}TOP OF MEMORY."
•40 PRINT"{CD}{CR}{RV}2.{RO} SPECI
FY START":PRINT"{CR}{CR}{CR}ADDRE
SS OF {RV}SOUNDER{RO}."
•50 INPUT"{CD}INPUT CHOICE";A
•60 IFA=1THENC=256*PEEK(56)+PEEK(5
5)-493:GOSUB260:CLR:GOTO130
•70 IFA=2THEN90
•80 GOTO10
•90 PRINT"{SC}{CR}STARTING ADDRESS
":INPUT"{CD}";C
•100 D=256*PEEK(44)+PEEK(43)+2760:
IFC<DTHEN90
•110 IFC>256*PEEK(56)+PEEK(55)-493
THEN90
•120 GOSUB260:CLR
•130 PRINT"{SC}{CD}{CD}{CD}{CD}{CD
}{CD}{CD}{CD}{CD}{CD}{CD}{CR}{CR}
{CR}{CR}{CR}READING DATA"
•140 A=256*PEEK(56)+PEEK(55)
•150 FORB=ATOA+492:READC:D=D+C
•160 PRINT"{HM}     {HM}";C
•170 POKEB,C:NEXT
•180 IFD<>49450THENPRINT"{SC}{CD}{
CR}ERROR IN DATA":END
•190 FORB=1TO9:READC:D=A+C:READE:F
=A+E
•200 POKED+1,INT(F/256):POKED,F-25
6*PEEK(D+1):NEXT
•210 B=A+376:POKEA+7,INT(B/256):PO
KEA+2,B-256*PEEK(A+7)
•220 PRINT"{SC}{CD}{RV}SOUNDER{RO}
 IN PLACE &":PRINT"{CD}TOP OF MEM
ORY LOWERED."
•230 PRINT"{CD}{CD}{CR}{RV}SYS{RO}
";A;"TO START"
•240 PRINT"{CD}{CD}{CR}{RV}SYS{RO}
";A+13;"TO STOP{CD}{CD}"
•250 NEW
•260 POKE56,INT(C/256):POKE55,C-25
6*PEEK(56):RETURN
•270 DATA120,169,108,141,20,3,169,
28,141,21
•280 DATA3,88,96,120,169,191,141,2
0,3,169
•290 DATA234,141,21,3,88,96,172,14
,3,200
•300 DATA208,2,230,1,177,0,96,160,
0,177
•310 DATA0,24,144,20,173,61,3,240,
12,172
•320 DATA77,3,206,61,3,32,17,27,24
,144
•330 DATA3,32,14,27,72,72,162,1,41
,224
•340 DATA208,6,142,68,3,24,144,55,
201,128
•350 DATA208,9,142,68,3,232,142,69
,3,208
•360 DATA42,201,192,208,6,142,68,3
,232,208
•370 DATA240,232,201,32,240,222,20
1,160,208,5
•380 DATA142,68,3,240,226,201,224,
208,6,142
•390 DATA68,3,232,208,216,232,201,
64,240,198
•400 DATA232,208,195,104,41,28,240
,62,162,1
•410 DATA142,70,3,201,4,208,8,142,
71,3
•420 DATA142,67,3,240,45,201,8,240
,247,201
•430 DATA12,208,5,142,73,3,240,238
,201,16
•440 DATA208,7,169,1,141,71,3,208,
21,201
•450 DATA20,240,17,201,28,208,10,1
42,64,3
•460 DATA169,128,141,63,3,208,3,14
2,73,3
•470 DATA104,41,3,240,18,201,1,208
,4,169
•480 DATA5,208,10,201,2,208,4,169,
10,208
•490 DATA2,169,15,141,76,3,32,17,2
```

```
7,72
•500 DATA41,127,208,4,169,1,208,1,
10,141
•510 DATA74,3,104,41,128,208,4,169
,10,208
•520 DATA2,169,22,141,65,3,32,17,2
7,72
•530 DATA41,127,24,105,128,174,68,
3,157,9
•540 DATA144,174,69,3,240,3,157,9,
144,104
•550 DATA41,128,141,75,3,173,14,14
4,41,240
•560 DATA24,109,76,3,141,14,144,14
0,77,3
•570 DATA96,172,66,3,208,69,189,9,
144,172
•580 DATA64,3,240,25,205,63,3,48,1
0,141
•590 DATA63,3,56,237,65,3,24,144,3
5,141
•600 DATA63,3,24,109,65,3,24,144,2
5,172
•610 DATA67,3,208,12,56,233,1,201,
127,208
•620 DATA13,238,66,3,208,8,189,9,1
44,24
•630 DATA105,1,240,243,157,9,144,1
74,69,3
•640 DATA240,3,157,9,144,96,173,62
,3,240
•650 DATA9,206,62,3,32,25,27,76,19
1,234
•660 DATA173,60,3,240,9,206,60,3,3
2,32
•670 DATA27,76,191,234,174,68,3,24
0,248,173
•680 DATA70,3,240,32,173,71,3,240,
13,173
•690 DATA72,3,240,5,206,72,3,240,1
4,238
•700 DATA72,3,32,33,28,173,73,3,24
0,6
•710 DATA174,68,3,32,33,28,206,74,
3,208
•720 DATA206,169,0,174,68,3,32,96,
28,173
•730 DATA75,3,208,8,238,60,3,238,6
1,3
•740 DATA208,8,173,14,144,41,240,1
41,14,144
•750 DATA169,0,162,15,157,61,3,202
,208,250
•760 DATA76,191,234
•770 DATA56,29,62,26,227,29,257,29
,385
•780 DATA37,399,44,433,301,444,301
,457,364
```



BUG REPELLENT LINE CODES FOR MAKER

```
LINE # 10:HD
LINE # 20:AP
LINE # 30:EA
LINE # 40:OA
LINE # 50:JE
LINE # 60:EK
LINE # 70:NP
LINE # 80:PH
LINE # 90:BK
LINE # 100:LK
LINE # 110:DM
LINE # 120:PH
LINE # 130:BN
LINE # 140:AF
LINE # 150:FA
LINE # 160:GO
LINE # 170:AI
LINE # 180:OJ
LINE # 190:GM
LINE # 200:OE
LINE # 210:DC
LINE # 220:AK
LINE # 230:DP
LINE # 240:FO
LINE # 250:KA
LINE # 260:PJ
LINE # 270:EC
LINE # 280:AI
LINE # 290:HE
LINE # 300:CG
LINE # 310:JP
LINE # 320:LB
LINE # 330:GO
LINE # 340:NI
LINE # 350:CB
LINE # 360:HF
LINE # 370:HD
LINE # 380:NK
LINE # 390:MP
LINE # 400:IE
LINE # 410:OB
LINE # 420:DG
LINE # 430:KG
LINE # 440:NJ
LINE # 450:OP
LINE # 460:AI
LINE # 470:JO
LINE # 480:DK
LINE # 490:NM
LINE # 500:GG
LINE # 510:NL
LINE # 520:FC
LINE # 530:KO
LINE # 540:CJ
LINE # 550:MK
LINE # 560:BM
LINE # 570:LA
LINE # 580:CI
LINE # 590:II
LINE # 600:PA
LINE # 610:JD
LINE # 620:FA
LINE # 630:OM
LINE # 640:GL
LINE # 650:FM
LINE # 660:KB
LINE # 670:MO
LINE # 680:AA
LINE # 690:IN
LINE # 700:HC
LINE # 710:BE
LINE # 720:GA
LINE # 730:ND
LINE # 740:OE
LINE # 750:BE
LINE # 760:IP
LINE # 770:OG
LINE # 780:HN
LINES: 78
```

IMPORTANT
Before typing in an *Ahoy!* program, refer to the first two pages of the program listings section.

```
2 Y=36879:POKE0,74:POKE1,3:GOTO24
4 PRINT"{CR}{RV}";M;"{RO}{CL}{CL}
{CL} {CR}{RV}.{RO} ";:RETURN
6 GOSUB14:IFPEEK(Y)=30THENPRINT"{
BL}";O$(2):GOTO10
8 POKEY,29:PRINT"{GN}";O$(1)
10 PRINT"{BK}{CD}":RETURN
12 INPUTY$:A=VAL(Y$):RETURN
14 PRINT"{SC}{CR}{CR}{CR}{CR}{CR}
{RV}";:RETURN
16 RESTORE:FORA=1TOM:READY$:NEXT:
FORA=1TOH:READY$:C$(A)=Y$:NEXT:RE
TURN
18 POKEY,26:PRINT"{SC}{CR}{CR}{RD
}{RV}";O$(4):POKE782,T:POKE828,1:
GOTO118
20 ONJGOSUB92,86,44,50,32,38,56:I
F(J=3ANDK=0)ORJ=7THENI=3:GOTO28
22 J=J+1:GOTO20
24 FORA=1TO6:POKE827+A,0:READY$:O
$(A)=Y$:NEXT:SYS6900
26 GOSUB130:IF(I=3ORI=4)ANDQ<3THE
N26
28 J=1:ONIGOSUB20,104,108,18,122,
138:IFI=2THENPOKEY,26:I=3:GOTO28
30 GOTO26
32 GOSUB6:M=18:H=8:GOSUB16:FORM=1
TO8:GOSUB4:PRINTC$(M):PRINT:NEXT:
PRINT"SELECT CHANNEL";:GOSUB12:B=
A
34 IFB<1ORB>8THEN32
36 D$(4)=C$(B):RETURN
38 GOSUB6:PRINT"SELECT STARTING",
"FREQUENCY VALUE{CD}","(128-255)"
;:GOSUB12:C=A:IFC<128ORC>255THEN1
38
42 D$(5)=STR$(C):RETURN
44 GOSUB6:PRINT"{CD}SELECT DURATI
ON{CD}","(2-254 JIFFIES)";:GOSUB1
2:D=A:D$(2)=STR$(D):IFD>254ORD<2T
HEN44
46 IFD>254ORD<2THEN44
48 RETURN
50 GOSUB6:M=26:H=9:GOSUB16:FORM=1
TO9:GOSUB4:PRINTC$(M):PRINT:NEXT
52 PRINT"SELECT MODIFIER";:GOSUB1
2:E=A:IFE<1ORE>9THEN50
54 D$(3)=C$(E):RETURN
56 A=2*C-512:IFE=1ORE>7THENRETURN
58 M=2^(E-2):F=INT(D+A/M):G=INT(-
A/M):U=INT(256-D*M/2)
60 IFE>4THENM=2^(E-5):F=INT(D-(A+
256)/M):G=INT((A+256)/M):U=INT(12
8+D*M/2)
62 IFF<1THENRETURN
64 GOSUB6:PRINT:M=11:H=3:GOSUB16:
PRINT"THE FACTORS WILL","CAUSE A
PAUSE OF",F;
66 PRINT"JIFFIES AFTER","THE SOUN
D":A=3:IFG<1THENC$(2)="":A=2
68 IF(E<5AND(U<129ORU>255))OR(E>4
AND(U>255ORU<129))THENC$(3)="":A=
A-1
70 PRINT"{CR}{CD}{RV}OPTIONS:{CD}
":FORM=1TOA:GOSUB4:PRINTC$(M);:IF
M=1THENPRINT,"{CR}{CD}{CD}{CD}{RV
}CHANGE:
72 IFM=2THENPRINTG
74 IFM=3THENPRINTU
76 PRINT:NEXT:IFM=2THENPRINT"{CD}
NO OTHER OPTIONS{CD}"
78 PRINT"INPUT CHOICE";:GOSUB12:I
FA<1ORA>M-1THEN64
80 IFA=2THEND$(2)=STR$(G):D=G
82 IFA=3THENC=U:D$(5)=STR$(U)
84 RETURN
86 GOSUB6:M=14:H=4:GOSUB16:FORM=1
TO4:GOSUB4:PRINTC$(M):PRINT:NEXT:
PRINT"SELECT VOLUME";
88 GOSUB12:K=A:IFK<1ORK>4THEN86
90 D$(1)=C$(K):K=K-1:RETURN
92 B=0:C=0:D=0:E=0:H=0:Q=Q+3:IFQ>
173THENGOSUB14:PRINT"{CD}TABLE FU
LL":GOTO118
94 IFQ=3THENT=Q:RETURN
96 GOSUB6:PRINT"START OF NEW{CD}"
,"SOUND STRING":INPUT"{CD}{CR}({R
V}Y{RO}/{RV}N{RO})";Y$:IFY$="Y"TH
ENT=Q
98 IFY$="N"ANDQ>3THENH=842+Q:IFPE
EK(H)>127THENPOKEH,PEEK(H)-128
100 IFY$="Y"ORY$="N"THENRETURN
102 GOTO96
104 POKEY,30:GOSUB6:GOSUB108:PRIN
T"{CD}CHANGE ITEM";:GOSUB12:IFA<1
ORA>5THEN104
106 ONAGOSUB86,44,50,32,38:GOSUB5
6:RETURN
108 M=6:H=5:GOSUB16:IFPEEK(Y)<>30
```

```
THENPOKEY,26:GOSUB14:PRINT"{RD}";
O$(3):PRINT"{BK}"
•110 IFK=0THENB=2:FORA=3TO5:D$(A)=
"":NEXT
•112 FORM=1TO5:GOSUB4:PRINTC$(M);D
$(M):PRINT:NEXT:IFPEEK(Y)=30THENR
ETURN
•114 B%(1)=K+4*E+32*B-36:B%(2)=INT
(D/2):IFE=9THENB%(2)=B%(2)+128:B%
(1)=B%(1)-4
•116 B%(3)=INT(C/2)+128:FORA=1TO3:
POKE842+Q+A,B%(A):NEXT:POKE782,Q:
POKE828,1
•118 FORA=1TO200:NEXT:IF(PEEK(Y-1)
AND15)>0ORPEEK(842)>0THEN118
•120 RETURN
•122 POKEY,24:GOSUB14:PRINT"{BK}";
O$(5):PRINT"{CD}{CD}FOR LAST STRI
NG:","{CD}{CD}DATA=";:FORA=843+TT
O845+Q
•124 PRINTPEEK(A);",";:NEXT:PRINT"
{CL} ":PRINT"{CD}{CD}{RV}AD{RO}DR
ESS=";843+T:PRINT"{CD}{CD}HIT KEY
 WHEN DONE
•126 GETY$:IFY$=""THEN126
•128 RETURN
•130 POKEY,31:PRINT"{SC}{CD}{CR}{B
L}{RV}SOUND{RO} {RV}CONCEPT{RO} {
RV}EDITOR{RO}{CD}
•132 PRINT"{BK}{CD}{CR}{RV}OPTIONS
:{CD}":FORM=1TO6:GOSUB4:PRINTO$(M
):IFM<>4THENPRINT
•134 NEXT:PRINT"{CD}INPUT CHOICE";
:GOSUB12:I=A:IFI<1ORI>6THEN130
•136 RETURN
•138 PRINT"{SC}":SYS6913
•140 DATACREATE SOUND,EDIT SOUND,P
LAY SOUND,PLAY SOUND STRING,DISPL
AY DATA,END
•142 DATAVOLUME ,DURATION,MOD ,CHA
N ,START FREQ,LEAVE AS IS,DURATIO
N TO,START FREQ TO
•144 DATAOFF (REST),LOW,MED,HIGH,S
1 (LOW),S2 (MED),S3 (HIGH),SN (NO
ISE),S1 & S2,S2 & S3
•146 DATAS1 & S3,S2 & SN,CONSTANT
FREQ,INCREASE SLOW,INCREASE MED,I
NCREASE FAST
•148 DATADECREASE SLOW,DECREASE ME
D,DECREASE FAST,WARBLE 1,WARBLE 2
```

BUG REPELLENT LINE CODES FOR EDITOR

```
LINE # 2:CI
LINE # 4:OK
LINE # 6:DA
LINE # 8:AG
LINE # 10:DA
LINE # 12:IO
LINE # 14:CC
LINE # 16:PP
LINE # 18:HB
LINE # 20:NF
LINE # 22:EG
LINE # 24:OM
LINE # 26:IL
LINE # 28:AB
LINE # 30:PE
LINE # 32:HG
LINE # 34:IB
LINE # 36:HJ
LINE # 38:CL
LINE # 42:LH
LINE # 44:DI
LINE # 46:PL
LINE # 48:IM
LINE # 50:AG
LINE # 52:NJ
LINE # 54:FB
LINE # 56:FL
LINE # 58:EJ
LINE # 60:BN
LINE # 62:PA
LINE # 64:HH
LINE # 66:GN
LINE # 68:DP
LINE # 70:FA
LINE # 72:EE
LINE # 74:FH
LINE # 76:BO
LINE # 78:CA
LINE # 80:PE
LINE # 82:AL
LINE # 84:IM
LINE # 86:MP
LINE # 88:HA
LINE # 90:AM
LINE # 92:EJ
LINE # 94:KN
LINE # 96:FO
LINE # 98:KG
LINE # 100:BK
LINE # 102:PJ
LINE # 104:PM
LINE # 106:BP
LINE # 108:IN
LINE # 110:GB
LINE # 112:IA
LINE # 114:CG
LINE # 116:IF
LINE # 118:FG
LINE # 120:IM
LINE # 122:LL
LINE # 124:BE
LINE # 126:LE
LINE # 128:IM
LINE # 130:KG
LINE # 132:GI
LINE # 134:OJ
LINE # 136:IM
LINE # 138:BP
LINE # 140:BI
LINE # 142:IF
LINE # 144:FL
LINE # 146:BG
LINE # 148:KL
LINES: 73
```

POST TIME

FROM PAGE 96

C-64 VERSION

```
•1 REM ****** P O S T  T I M E ***
***
•2 REM *** DESIGNED BY BOB LLORET
***
•3 REM ***** FOR AHOY! MAGAZINE **
***
•5 PRINT "{SC}":POKE 53280,6:POKE
```

```
53281,0:FOR A=1 TO 3:PRINT:NEXT A
10 PRINT TAB(6);"{YL}{RV}   *{RO}
 {RV}£  *{RO} {RV}£  *{RO} {RV}£
  *"
•15 PRINT TAB(6);"{GN}{RV} {RO}  {
RV} {RO} {RV} {RO}  {RV} {RO} {RV
} {RO}       {RV} {RO}       £ £ £"
•20 PRINT TAB(6);"{BL}{RV}    {RO}£
{RO} {RV} {RO}  {RV} {RO} *{RV}
*{RO}   {RV} {RO}        £ £ £"
•25 PRINT TAB(6);"{PU}{RV} {RO}
 {RV} {RO}  {RV} {RO}     {RV} {RO
}   {RV} {RO}       £ £ £"
•30 PRINT TAB(6);"{RD}{RV} {RO}
 *{RV}  {RO}£{RO} *{RV}   {RO}£
{RV} "
•32 PRINT
•35 PRINT TAB(16);"{BL}{RV}£   *{R
O} {RV}£ *{RO} {RV}*{RO}  {RV}£{R
O} {RV}£  *"
•40 PRINT TAB(6);"{GN}£ £ £      {G
N}  {RV} {RO}    {RV} {RO}  {RV}
*£ {RO} {RV} "
•45 PRINT TAB(6);"{YL}£ £ £
{RV} {RO}     {RV} {RO}  {RV} {RO}
   {RV} {RO} {RV}  "
•50 PRINT TAB(6);"{RD}£ £ £
{RV} {RO}     {RV} {RO}  {RV} {RO}
   {RV} {RO} {RV} "
•55 PRINT TAB(16);"{PU}  {RV} {RO}
   *{RV} {RO}£{RO} {RV} {RO}   {RV
} {RO} *{RV}  {RO}£"
•60 PRINT "{CD}{CD}{CD}{CD}{CD}":M
SG$="                       DESIGNE
D BY....BOB LLORET"
•65 PRINT " {BL}{RV}         {RO}
                   {RV}          "
•70 A$=RIGHT$(MSG$,LEN(MSG$)-1):B$
=A$+LEFT$(MSG$,1)
•75 PRINT "{PU}{CU}";TAB(7);LEFT$(
B$,26):MSG$=B$
•80 FOR D=1 TO 80:NEXT D:MSG=MSG+1
:IF MSG=75 THEN 90
•85 GOTO 70
•90 A$="":B$="":MSG$="":MSG=0
•100 PRINT "{SC}":POKE 53280,6:POK
E 53281,7
•110 PRINT "{BL}{RV}{CU}
                                 "
•120 PRINT "{BL}{RV}{CU}
                                 "
•130 PRINT "{RV}{CU} *CCCCCCC*  P
O S T  T I M E{RV}  *CCCCCCC*  "
•140 PRINT "{BL}{RV}{CU}
                                 "
•160 PRINT "{BL}{RV}{CU}
                                 "
•170 PRINT TAB(10);"{PU}{CD}NUMBER
 OF PLAYERS";:INPUT NP:IF NP<1 OR
 NP>4 THEN 100
•180 PRINT "{BR}{CD}        ENTER P
LAYER'S FIRST NAMES":PRINT
•185 FOR I=1 TO NP:PRINT "{RD}{CD}
          PLAYER #";I;:INPUT NAME
$(I):NEXT I
•190 PRINT "{SC}":POKE 53280,0:POK
E 53281,0
•195 PRINT TAB(7);"{LB}{CD}{CD}WEL
COME TO COMMODORE DOWNS"
•200 PRINT "{GN}{CD}{CD}{CD}{RV}[1
]{RO} ALL PLAYERS START WITH $500
•205 PRINT "{RV}{CD}{CD}[2]{RO} AL
L PLAYERS WILL BET IN TURN"
•210 PRINT "{RV}{CD}{CD}[3]{RO} TH
E PROGRAM WILL KEEP TRACK OF WINS
     AND LOSSES"
•220 PRINT "{RD}{RV}{CD}{CD}[4]{RO
} IF YOU LOSE YOUR MONEY, YOU WIL
L NOT    BE ABLE TO BET"
•298 REM ****  CREATE SPRITES  ***
*
•299 REM =========================
=
•300 V=53248
•310 FOR SP=12288 TO 12350:READ NU
:POKE SP,NU:NEXT SP
•320 FOR SP=12352 TO 12414:READ NU
:POKE SP,NU:NEXT SP
•330 FOR SP=12416 TO 12478:READ NU
:POKE SP,NU:NEXT SP
•340 FOR SP=12480 TO 12542:READ NU
:POKE SP,NU:NEXT SP
•345 FOR SP=12544 TO 12606:READ NU
:POKE SP,NU:NEXT SP
•350 DATA 0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,14,0,1,252,128,1,248,192
•360 DATA 32,229,224,119,255,240,2
07,254,96,143,254,0,7,252,0,6,12
•370 DATA 0,6,12,0,3,24,0,0,144,0,
0,0,0,0,0,0,0,0,0,0,0,0
•380 DATA 0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,14,0,1,252,64,0,228,96,0,
0,240
•390 DATA 231,255,248,127,255,176,
15,254,0,15,254,0,12,6,0,12,3,0
•400 DATA 6,1,128,2,3,0,0,0,0,0,0,
```

```
0,0,0,0,0,0,0
•410 DATA 0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,14,0,1,252,0,1,248,
32,224
•420 DATA 230,112,55,255,248,15,25
5,188,15,255,24,15,254,0,12,6,0,1
2,3
•430 DATA 0,24,3,0,16,1,128,0,0,0,
0,0,0,0,0,0
•440 DATA 0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,14,0,1,252,0,225,25
2,16,112
•450 DATA 227,48,23,255,248,15,255
,252,15,255,14,15,254,4,12,6,0
•455 DATA 24,3,0,48,1,128,96,0,192
,0,0,0,0,0,0,0,0,0
•460 DATA 0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,56,0,48,60,0,248
•461 DATA 122,7,248,254,63,252,254
,127,255,254,127,255,254,127,255,
254
•462 DATA 255,255,255,255,255,255,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
•465 PRINT "{CD}{PU}     IF YOUR RE
ADY FOR SOME EXCITEMENT...     ...
PRESS 'F1'"
•466 GET AN$:IF AN$="{F1}" THEN 47
0
•467 IF AN$="" THEN 466
•468 REM ***  RACE SELECTIONS  ***
•469 REM ========================
•470 FOR I=1 TO NP:CASH(I)=500:NEX
T I:BR=0:B=0:TS=0:FP=0:HP=0:SI=54
272
•475 CNT=CNT+1:FOR L=0 TO 24:POKE
SI+L,0:NEXT L
•480 PRINT "{SC}":POKE 53280,9:POK
E 53281,7
•490 PRINT "{BR}{RV}{CU}BBBBBBBBB{
RO}                     {RV}BBBBB
BBBBB"
•500 PRINT "{RV}{CU}BBBBBBBBB{RO}{
BL} RACE #";CNT;"SELECTIONS {RV}{
BR}BBBBBBBBBBBB"
•510 PRINT "{RV}{CU}BBBBBBBBB{RO}F
FFFFFFFFFFFFFFFFFFFF{RV}BBBBBBBBB
B"
•520 PRINT:PRINT "{BL} NO.
NAME                 ODDS "
•530 FOR I=1 TO 6:T=31:OD(I)=INT(R
ND(0)*15)+1:READHNAME$(I):IFOD(I)
>9 THEN T=30
•540 PRINT "{BK} ";I;TAB(9);"{PU}"
```

```
;HNAME$(I);TAB(T);"{RD}";OD(I);"T
O 1":NEXT I
•550 PRINT:PRINT "{BL}    $ $ $ {BR
}{RV}TRIFECTA PAYS 50 TO 1{BL}{RO
} $ $ $":PRINT
•555 FOR I=1 TO NP:IF CASH(I)<=0 T
HEN BET(I)=0:H(I)=0:GOTO 695
•560 PRINT "{G1}           OKAY ";NAM
E$(I);", MAKE YOUR BET"
•570 PRINT:PRINTTAB(15);"{LB}1-WIN
":PRINTTAB(15);"2-PLACE":PRINTTAB
(15);"3-SHOW"
•580 PRINT TAB(15);"{RD}4-TRIFECTA
"
•590 GET AN$(I):IF AN$(I)="" THEN
590
•600 IF AN$(I)<"1" OR AN$(I)>"4" T
HEN PRINT "{CU}{CU}{CU}{CU}{CU}{C
U}{CU}":GOTO 560
•610 PRINT"{CU}{CU}{CU}{CU}{CU}{CU
}":FORA=1TO7:PRINT"
                            ":NEXT
•615 PRINT "{CU}{CU}{CU}{CU}{CU}{C
U}{CU}"
•620 IF AN$(I)>="1" AND AN$(I)<="3
" THEN GOTO 640
•630 IF AN$(I)="4" THEN GOTO 670
•640 PRINT TAB(12);"{CU}{LB}WHICH
HORSE";:INPUT H(I):IF H(I)<1 OR H
(I)>6 THEN 640
•650 PRINT TAB(5);"{CD}YOU HAVE $"
;CASH(I);" YOUR BET";:INPUT BET(I
)
•660 IF BET(I)>CASH(I) THEN PRINT
"{CU}{CU}{CU}":GOTO 650
•665 PRINT "{CD}":GOTO 685
•670 FOR TRI=1 TO 3:PRINT TAB(10);
"{CU}HORSE NUMBER ";TRI;:INPUT H(
TRI)
•675 PRINT:NEXT TRI
•680 PRINT TAB(5);"YOU HAVE $";CAS
H(I);" YOUR BET";:INPUT BET(I)
•682 IF BET(I)>CASH(I) THEN PRINT
"{CU}{CU}{CU}":GOTO 680
•685 PRINT"{CU}{CU}{CU}{CU}{CU}{CU
}{CU}":FORA=1TO7:PRINT"
                             ":NEX
T
•690 PRINT "{CU}{CU}{CU}{CU}{CU}{C
U}{CU}{CU}"
•695 NEXT I
•698 REM ****  THE RACE  ****
•699 REM ===================
```

```
•700 GOSUB 2000
•705 POKE V+16,0:B=0:TS=0:FP=0:HP=
0
•710 P=30:PY=135:FOR N=1 TO 6:X(N)
=P:Y(N)=PY:PY=PY+14:NEXT N
•720 POKEV+39,1:POKEV+40,14:POKEV+
41,7:POKEV+42,3:POKEV+43,4:POKE V
+44,0
•730 A=0:FOR N=1TO 6:POKE V+A,X(N)
:A=A+1:POKE V+A,Y(N):A=A+1:NEXT N
•735 FOR N=1 TO 6:COL(N)=30:NEXT N
•740 FOR S=2040 TO 2045:POKE S,192
:NEXT S:POKE V+21,255
•750 PRINT "{BK}{CU}             T H E
Y' R E   O F F ! ! !"
•760 FOR D=1 TO 1000:NEXT D
•770 PRINT "{CU}
                   "
•815 REM ***  HORSE  MOVEMENT  ***
•816 REM ==========================
•820 Z=INT(RND(0)*6)+1
•830 IF Z=1 THEN A=0:B=1:R=1:N=1:H
=2040
•840 IF Z=2 THEN A=2:B=2:R=3:N=2:H
=2041
•850 IF Z=3 THEN A=4:B=4:R=5:N=3:H
=2042
•860 IF Z=4 THEN A=6:B=8:R=7:N=4:H
=2043
•870 IF Z=5 THEN A=8:B=16:R=9:N=5:
H=2044
•880 IF Z=6 THEN A=10:B=32:R=11:N=
6:H=2045
•890 Y(N)=Y(N)-3:POKE V+R,Y(N):POK
EH,193:FOR D=1 TO 4:NEXT D:POKEH,
194
•895 FOR D=1 TO 4:NEXT D:X(N)=X(N)
+5:COL(N)=COL(N)+5
•896 POKE SI,2:POKE SI+1,1:POKESI+
24,8:POKE SI+4,129
•900 IF X(N)>=255 THEN X(N)=0:TS=T
S+B:POKE V+16,TS
•905 IF COL(N)=155 THEN GOSUB 1100
•910 IF COL(N)=320 THEN GOSUB 935
•912 POKE SI+4,16:POKE SI,0:POKE S
I+1,0
•913 FOR D=1 TO 3:NEXT D
•915 POKE SI,2:POKE SI+1,1:POKESI+
24,8:POKE SI+4,129
•920 POKEV+A,X(N):Y(N)=Y(N)+3:POKE
V+R,Y(N):POKEH,195:FORD=1TO10:NEX
TD:POKEH,192
•925 POKE SI+4,16:POKE SI,0:POKE S
I+1,0
•930 GOTO 820
•935 FP=FP+1:WIN(FP)=N:WIN$(FP)=HN
AME$(N):OD(FP)=OD(N)
•940 IF FP=3 THEN POKE SI+4,32:POK
E SI+1,0:POKE SI,0:GOTO 950
•945 RETURN
•948 REM **** PHOTO FINISH ****
•949 REM ======================
•950 PRINT "{SC}":POKE V+21,0:POKE
 53280,1:POKE 53281,1:PRINT "{CD}
{CD}{CD}{CD}{CD}{CD}"
•955 FOR D=1 TO 600:NEXT D
•960 PRINT TAB(15);"{G3}P H O T O"
:PRINT TAB(14);"{CD}{CD}{CD}F I N
 I S H"
•970 FOR D=1 TO 500:NEXT D
•980 PRINT TAB(15);"{G2}{CU}{CU}{C
U}{CU}{CU}P H O T O":PRINT TAB(14
);"{CD}{CD}{CD}F I N I S H"
•990 FOR D=1 TO 500:NEXT D
•1000 PRINT TAB(15);"{G1}{CU}{CU}{
CU}{CU}{CU}P H O T O":PRINT TAB(1
4);"{CD}{CD}{CD}F I N I S H"
•1010 FOR D=1 TO 500:NEXT D
•1020 PRINT TAB(15);"{CU}{CU}{CU}{
CU}{CU}{BK}P H O T O":PRINT TAB(1
4);"{CD}{CD}{CD}F I N I S H"
•1030 FOR D=1 TO 700:NEXT D:GOTO 1
150
•1100 HP=HP+1:PLA(HP)=N:IF HP=3 TH
EN 1115
•1110 GOTO 1120
•1115 PRINTTAB(4);"{BL}{CU}AT THE
1/2 MILE POLE-";PLA(1);" ";PLA(2)
;" ";PLA(3);
•1120 RETURN
•1140 REM **** RACE RESULTS ****
•1145 REM ======================
•1150 PRINT "{SC}":POKE 53280,2:PO
KE 53281,15
•1160 PRINT TAB(6);"{RD}**{BL} R A
 C E  R E S U L T S{RD} **":PRINT
•1170 PRINT TAB(10);"{BL}{CD}WIN #
"
•1180 PRINT TAB(8);"{CD}PLACE #"
•1190 PRINT TAB(9);"{CD}SHOW #":PR
INT "{CU}{CU}{CU}{CU}{CU}{CU}{CU}
"
•1195 FOR D=1 TO 1000:NEXT D
•1200 FOR FP=1 TO 3:PRINT TAB(15);
"{BK}{CD}";WIN(FP);"{PU} ";WIN$(F
P)
```

```
•1205 FOR D=1 TO 1300:NEXT D:NEXT
 FP
•1210 PRINT "{BK}££££££££££££££££££
££££££££££££££££££££££££"
•1220 FOR I=1 TO NP
•1240 IF AN$(I)="1" THEN 1280
•1250 IF AN$(I)="2" THEN 1300
•1260 IF AN$(I)="3" THEN 1320
•1270 IF AN$(I)="4" THEN 1340
•1280 IF H(I)=WIN(1) THEN AMT(I)=B
 ET(I)*OD(1):GOTO 1360
•1290 CASH(I)=CASH(I)-BET(I):GOTO
 1380
•1300 IF H(I)=WIN(1) THEN AMT(I)=I
 NT(BET(I)*OD(1)/2):GOTO 1360
•1305 IF H(I)=WIN(2) THEN AMT(I)=I
 NT(BET(I)*OD(2)/2):GOTO 1360
•1310 CASH(I)=CASH(I)-BET(I):GOTO
 1380
•1320 IF H(I)=WIN(1) THEN AMT(I)=I
 NT(BET(I)*OD(1)/3:GOTO 1360
•1325 IF H(I)=WIN(2) THEN AMT(I)=I
 NT(BET(I)*OD(2)/3):GOTO 1360
•1326 IF H(I)=WIN(3) THEN AMT(I)=I
 NT(BET(I)*OD(3)/3):GOTO 1360
•1330 CASH(I)=CASH(I)-BET(I):GOTO
 1380
•1340 IF H(1)=WIN(1)ANDH(2)=WIN(2)
  AND H(3)=WIN(3)THENAMT(I)=BET(I)
 *50:GOTO1360
•1350 CASH(I)=CASH(I)-BET(I):GOTO
 1380
•1360 CASH(I)=CASH(I)+AMT(I)
•1370 PRINT "{BL}{CD}";TAB(5);NAME
 $(I);" WON $";AMT(I);"-YOU HAVE $
 ";CASH(I):GOTO 1500
•1380 IF CASH(I)<=0 THEN 1400
•1390 PRINT"{RD}{CD}";TAB(5);NAME$
 (I);" LOST $";BET(I);"-YOU HAVE $
 ";CASH(I):GOTO 1500
•1400 PRINT "{BK}{CD}";TAB(5);NAME
 $(I);" WENT BROKE":BR=BR+1:IF BR=
 NP THEN END
•1500 NEXT I
•1510 IF CNT=10 THEN END
•1520 IF NP=1 THEN LINE=9
•1530 IF NP=2 THEN LINE=7
•1540 IF NP=3 THEN LINE=5
•1550 IF NP=4 THEN LINE=3
•1560 FOR A=1 TO LINE:PRINT:NEXT A
•1570 PRINT " {BK}{RV}          {RO}{
 RD} NEXT RACE....COMING UP {RV}{B
 K}       "
•1580 FOR D=1 TO 5500:NEXT D:GOTO
 475
•1998 REM *** DRAW RACE TRACK ***
•1999 REM =======================
•2000 POKE V+45,1:POKE V+12,85:POK
 E V+13,48:POKE 2046,196:POKE V+29
 ,192
•2010 POKE V+21,192:POKE 2047,196:
 POKE V+46,1:POKE V+14,120:POKE V+
 15,48
•2020 PRINT "{SC}":POKE 53280,0:PO
 KE 53281,5
•2030 FOR A=1 TO 3:PRINT "{BL}{CU}
 {RV}
             ":NEXT A
•2040 PRINTTAB(9);"{G1}*{RD}£    {G
 1}{RO}*{YL}£    {G1}*{BL}£    {G1}*
 {OR}£    {G1}*{PU}£"
•2050 PRINT TAB(9);"{G1}*   {BK}COM
 MODORE  DOWNS{G1} *"
•2060 PRINT TAB(10);"{G2}*{RV}
              {RO}£"
•2070 PRINT TAB(11);"{G2}*{RV}{WH}
 ++++++++++++++ {RO}{G2}£"
•2080 PRINT TAB(12);"{WH}{RV} ++++
 ++++++++++ ":PRINT TAB(12);"{G1}{
 RV}                  "
•2090 PRINT "{WH}POYYPOYYPOYYPOYYP
 OYYPOYYPOYYPOYYPOYYPOYY"
•2100 PRINT "{CU}{BK}{RV}1{BR}
            {38 SPACES}
 {BK}F"
•2110 PRINT "{CU}{BK}{RV} {BR}
            {38 SPACES}
 {BK} "
•2120 PRINT "{CU}{BK}{RV}2{BR}
            {38 SPACES}
 {BK}I"
•2130 PRINT "{CU}{BK}{RV} {BR}
            {38 SPACES}
 {BK} "
•2140 PRINT "{CU}{BK}{RV}3{BR}
            {38 SPACES}
 {BK}N"
•2150 PRINT "{CU}{BK}{RV} {BR}
            {38 SPACES}
 {BK} "
•2160 PRINT "{CU}{BK}{RV}4{BR}
            {38 SPACES}
 {BK}I"
•2170 PRINT "{CU}{BK}{RV} {BR}
            {38 SPACES}
 {BK} "
```

```
•2180 PRINT "{CU}{BK}{RV}5{BR}
              {38 SPACES}
  {BK}S"
•2190 PRINT "{CU}{BK}{RV} {BR}
              {38 SPACES}
  {BK} "
•2200 PRINT "{CU}{BK}{RV}6{BR}
              {38 SPACES}
  {BK}H"
•2210 PRINT "{WH}{CU}POYYPOYYPOYYP
OYYPOYYPOYYPOYYPOYYPOYYPOYY"
•2220 RETURN
•2300 DATA SEATTLE SLEW,SECRETARIA
T,LIGHTNIN',SLO POKE,STREAKER,GAM
BLER
•2310 DATA FIREWORKS,HOOFER,THUNDE
R,SLICK STICK,GOLDEN BOY,ROCKET
•2320 DATA BOLD FORBES,AFFIRMED,GE
NERAL ASSEMBLY,DANCERS IMAGE,HILL
 RISE,NASHUA
•2330 DATA GENUINE RISK,RIVA RIDGE
,PENSIVE,ASSAULT,TIM TAM,CARRY BA
CK
•2340 DATA JIM FRENCH,NO LE HACE,B
REVITY,NASHUA,FABIUS,ASSAULT
•2350 DATA MAJESTIC PRINCE,PENSIVE
,WORTH,OMAHA,JET PILOT,COUNT TURF
•2360 DATA DAUBER,CROZIER,SHAM,CAR
RY BACK,PONDER,POT O LUCK
•2370 DATA ADVOCATOR,HILL PRINCE,S
PY SONG,MISSTEP,ZAL,TICKET
•2380 DATA BALLY ACHE,CAPOT,CITATI
ON,NEEDLES,DECIDEDLY,SWAPS
•2390 DATA FOWARD PASS,RUMBO,BURGO
O KING,CAVALCADE,FLYING EBONY,NAT
IVE DANCER
```

BUG REPELLENT LINE CODES
FOR POST TIME (C-64)

```
LINE # 1:OE
LINE # 2:DE
LINE # 3:MF
LINE # 5:IJ
LINE # 10:KB
LINE # 15:IP
LINE # 20:JG
LINE # 25:MM
LINE # 30:CF
LINE # 32:JJ
LINE # 35:EB
LINE # 40:OB
LINE # 45:KG
LINE # 50:FM
LINE # 55:OG
LINE # 60:HA
LINE # 65:FA
LINE # 70:JA
LINE # 75:CF
LINE # 80:CK
LINE # 85:PF
LINE # 90:BH
LINE # 100:OO
LINE # 110:OJ
LINE # 120:OJ
LINE # 130:DC
LINE # 140:OJ
LINE # 160:OJ
LINE # 170:AI
LINE # 180:ML
LINE # 185:ON
LINE # 190:OD
LINE # 195:NA
LINE # 200:AM
LINE # 205:KH
LINE # 210:JF
LINE # 220:HA
LINE # 298:KO
LINE # 299:DB
LINE # 300:AD
LINE # 310:FB
LINE # 320:BM
LINE # 330:PJ
LINE # 340:LA
LINE # 345:HG
LINE # 350:PF
LINE # 360:AC
LINE # 370:NJ
LINE # 380:AN
LINE # 390:PP
LINE # 400:DE
LINE # 410:BD
LINE # 420:BK
LINE # 430:GA
LINE # 440:GD
LINE # 450:GP
LINE # 455:PH
LINE # 460:BF
LINE # 461:ON
LINE # 462:FD
LINE # 465:BE
LINE # 466:ED
LINE # 467:AE
LINE # 468:EC
LINE # 469:OO
LINE # 470:KG
LINE # 475:DG
LINE # 480:GE
LINE # 490:EA
LINE # 500:BJ
LINE # 510:LA
LINE # 520:PB
LINE # 530:BH
LINE # 540:GN
LINE # 550:CP
LINE # 555:OD
LINE # 560:LG
LINE # 570:EF
LINE # 580:GA
LINE # 590:BB
LINE # 600:EG
LINE # 610:ME
LINE # 615:PC
LINE # 620:PH
LINE # 630:HG
LINE # 640:HC
LINE # 650:LC
LINE # 660:JJ
LINE # 665:LG
LINE # 670:MA
LINE # 675:FA
LINE # 680:ON
LINE # 682:JE
LINE # 685:EC
LINE # 690:JD
LINE # 695:ON
LINE # 698:MJ
LINE # 699:LE
LINE # 700:HE
LINE # 705:JL
LINE # 710:LC
LINE # 720:LH
LINE # 730:JN
LINE # 735:KL
LINE # 740:CA
LINE # 750:EA
LINE # 760:JO
LINE # 770:JL
LINE # 815:FB
LINE # 816:OO
LINE # 820:BD
LINE # 830:EO
LINE # 840:EI
LINE # 850:HN
LINE # 860:JM
LINE # 870:OL
LINE # 880:GN
LINE # 890:CL
LINE # 895:JE
LINE # 896:NO
LINE # 900:FM
LINE # 905:MD
LINE # 910:JP
LINE # 912:KC
LINE # 913:CD
LINE # 915:NO
LINE # 920:LC
LINE # 925:KC
LINE # 930:EC
LINE # 935:MD
LINE # 940:CN
LINE # 945:IM
LINE # 948:JN
LINE # 949:DF
```

```
LINE # 950:OJ
LINE # 955:JP
LINE # 960:ON
LINE # 970:GC
LINE # 980:NK
LINE # 990:GC
LINE # 1000:KJ
LINE # 1010:GC
LINE # 1020:DA
LINE # 1030:PO
LINE # 1100:KG
LINE # 1110:HE
LINE # 1115:FB
LINE # 1120:IM
LINE # 1140:JD
LINE # 1145:DF
LINE # 1150:GC
LINE # 1160:NC
LINE # 1170:NC
LINE # 1180:NE
LINE # 1190:ID
LINE # 1195:JO
LINE # 1200:CD
LINE # 1205:DD
LINE # 1210:CE
LINE # 1220:FO
LINE # 1240:CJ
LINE # 1250:CB
LINE # 1260:CO
LINE # 1270:CL
LINE # 1280:FF
LINE # 1290:CA
LINE # 1300:GD
LINE # 1305:DN
LINE # 1310:CA
LINE # 1320:AG
LINE # 1325:OA
LINE # 1326:CC
LINE # 1330:CA
LINE # 1340:AM
LINE # 1350:CA
LINE # 1360:EG
LINE # 1370:DA
LINE # 1380:IH
LINE # 1390:JB
LINE # 1400:EB
LINE # 1500:ON
LINE # 1510:CA
LINE # 1520:AA
LINE # 1530:DB
LINE # 1540:DG
LINE # 1550:DH
LINE # 1560:IF
LINE # 1570:GB
LINE # 1580:EG
LINE # 1998:NO
LINE # 1999:GJ
LINE # 2000:IN
LINE # 2010:KK
LINE # 2020:GB
LINE # 2030:PF
LINE # 2040:DI
LINE # 2050:MB
LINE # 2060:DL
LINE # 2070:AA
LINE # 2080:KL
LINE # 2090:PG
LINE # 2100:MP
LINE # 2110:FJ
LINE # 2120:OL
LINE # 2130:FJ
LINE # 2140:OF
LINE # 2150:FJ
LINE # 2160:OJ
LINE # 2170:FJ
LINE # 2180:OC
LINE # 2190:FJ
LINE # 2200:NA
LINE # 2210:CF
LINE # 2220:IM
LINE # 2300:KC
LINE # 2310:IH
LINE # 2320:HJ
LINE # 2330:JJ
LINE # 2340:AA
LINE # 2350:DD
LINE # 2360:DO
LINE # 2370:DL
LINE # 2380:PM
LINE # 2390:GP
LINES: 224
```

VIC 20 VERSION

IMPORTANT

Before typing in an *Ahoy!* program, refer to the first two pages of the program listings section.

```
•5 PRINT"{SC}{CD}{CD}{CD}":POKE368
79,14
•10 PRINT"{CR}{YL}{RV}    *{RO} {RV
}£  *{RO} {RV}£  *{RO} {RV})   *"
•15 PRINT"{CR}{GN}{RV} {RO}   {RV}
{RO} {RV} {RO}  {RV} {RO} {RV} {R
O}        {RV} {RO}"
•20 PRINT"{CR}{BL}{RV}    {RO}){RO}
 {RV} {RO}  {RV} {RO} *{RV}  *{RO
}    {RV} {RO}"
•25 PRINT"{CR}{PU}{RV} {RO}     {RV
} {RO}  {RV} {RO}     {RV} {RO}
{RV} {RO}"
•30 PRINT"{CR}{RD}{RV} {RO}      *{R
V}  {RO}£{RO} *{RV}  {RO}£   {RV}
 "
•35 PRINT"{CR}{CR}{CD}{BL}{RV}£
*{RO} {RV}£ *{RO} {RV}*{RO}  {RV}
£{RO} {RV}£   *"
•40 PRINT"{CR}{CR}{GN}  {RV} {RO}
   {RV} {RO}  {RV} *£ {RO} {RV} "
•45 PRINT"{CR}{CR}{YL}  {RV} {RO}
   {RV} {RO}  {RV} {RO}  {RV} {RO
} {RV} "
•50 PRINT"{CR}{CR}{RD}  {RV} {RO}
   {RV} {RO}  {RV} {RO}  {RV} {RO
} {RV} "
•55 PRINT"{CR}{CR}{PU}  {RV} {RO}
  *{RV} {RO}£{RO} {RV} {RO}  {RV}
 {RO} *{RV}  {RO}£"
•61 A$=" GAME BY....BOB LLORET "
•62 B$="VERSION BY ROB ALONSO "
•63 FORX=1TO45
•64 A$=RIGHT$(A$,21)+LEFT$(A$,1)
•65 B$=RIGHT$(B$,1)+LEFT$(B$,21):F
ORD=1TO80:NEXT
•66 PRINT"{HM}{CD}{CD}{CD}{CD}{CD}
{CD}{CD}{CD}{CD}{CD}{CD}{CD}{CD}{
CD}{CD}{CD}{CD}{WH}";A$;"{CY}{CD}
";B$:NEXT
•70 FORI=ITO500:NEXT:PRINT"{SC}{BL
}":POKE36879,126
•110 A$=" {RV}
"
•120 PRINTA$:PRINTA$
•130 PRINT" {RV}  *C* POST TIME *C
* "
•140 PRINTA$:PRINTA$
•180 INPUT"{PU}{CD}{CD}{CR}PLAYER'
S NAME";NAME$
•190 PRINT"{SC}":POKE36879,8
```

```
•195 PRINT"{RV}{WH}{CR}WELCOME TO
COMMODORE{RO}              {RV}DOWNS"
•200 PRINT "{GN}{RV}{CD}[1]{RO} YO
U WILL BEGIN          WITH $500
•205 PRINT "{RV}{CD}{CD}[2]{RO} YO
U CAN PLACE A         BET EACH TURN
"
•210 PRINT "{RV}{CD}{CD}[3]{RO} TH
E PROGRAM WILL        KEEP TRACK OF
 WINS    AND LOSSES"
•220 PRINT "{RD}{RV}{CD}{CD}[4]{RO
} IF YOU LOSE YOUR          MONEY, TH
E GAME       ENDS"
•230 PRINT "{CD}{PU}  PRESS 'F1' T
O PLAY"
•466 GETAN$:IFAN$<>"{F1}"THEN466
•470 CASH=500:CNT=CNT+1
•480 PRINT"{SC}":POKE36879,123
•490 PRINT"{WH}{RV}{CU}BBBBBB{RO}
          {RV}BBBBBB"
•500 PRINT"{RV}{CU}BBBBBB{RO}{BL}
RACE #";CNT;"{RV}{WH}BBBBBB"RDOWN
S"
•510 PRINT "{RV}{CU}BBBB{RO}FFFFFF
FFFFFFF{RV}BBBBB"   WITH $500
•520 PRINT"{CD}{BL}NO. NAME
ODDS{CD}"
•530 FORI=1TO6:OD(I)=INT(RND(0)*15
)+1:READHNAME$(I)
•540 PRINT"{WH}";I;"{PU} ";HNAME$(
I)
•543 A=LEN(STR$(OD(I)))+5:PRINT"{R
D}{CU}";TAB(21-A);OD(I);"TO 1":NE
XT
•550 PRINT"{CD}{BL}{RV}
          "
•560 PRINT"{PU}{CU}OKAY ";NAME$:PR
INT"MAKE YOUR BET"
•640 INPUT"{GN}WHICH HORSE";H:IFH<
1ORH>6THENPRINT"{CU}{CU}":GOTO640
•650 PRINT"YOU HAVE $";CASH
•655 INPUT"YOUR BET";BET
•660 IFBET>CASHTHENPRINT"{CU}{CU}{
CU}":GOTO650
•700 GOSUB2020
•710 FORX=1TO6:HO(X)=0:NEXT
•750 PRINT "{WH}{HM}T H E Y' R E
O F F !"
•810 POKE36878,0:A$="{CR}{CR}{CR}{
CR}{CR}{CR}{CR}{CR}{CR}{CR}{CR}{C
R}{CR}{CR}{CR}{CR}{CR}{CR}{CR}{CR
}{CR}{CR}"
•820 Z=INT(RND(0)*6)+1
•830 IFZ=1THENHO(1)=HO(1)+1:PRINT"
{HM}{CD}{CD}{CD}{CD}{CD}{CD}{CD}{
CD}{CD}"RIGHT$(A$,HO(1))+"{CL}{RV
"
•840 IFZ=2THENHO(2)=HO(2)+1:PRINT"
{HM}{CD}{CD}{CD}{CD}{CD}{CD}{CD}{
CD}{CD}{CD}{CD}"RIGHT$(A$,HO(2))+
"
•850 IFZ=3THENHO(3)=HO(3)+1:PRINT"
{HM}{CD}{CD}{CD}{CD}{CD}{CD}{CD}{
CD}{CD}{CD}{CD}{CD}{CD}"RIGHT$(A$
"
•860 IFZ=4THENHO(4)=HO(4)+1:PRINT"
{HM}{CD}{CD}{CD}{CD}{CD}{CD}{CD}{
CD}{CD}{CD}{CD}{CD}{CD}{CD}{CD}"R
"
•870 IFZ=5THENHO(5)=HO(5)+1:PRINT"
{HM}{CD}{CD}{CD}{CD}{CD}{CD}{CD}{
CD}{CD}{CD}{CD}{CD}{CD}{CD}{CD}{C
D}{CD}"RIGHT$(A$,HO(5))+"{CL}{RV}
"
•880 IFZ=6THENHO(6)=HO(6)+1:PRINT"
{HM}{CD}{CD}{CD}{CD}{CD}{CD}{CD}{
CD}{CD}{CD}{CD}{CD}{CD}{CD}{CD}{C
D}{CD}{CD}{CD}"RIGHT$(A$,HO(6))+"
"
•890 FORX=1TO6:IFHO(X)>20THENGOTO1
000
•900 NEXT:POKE36878,8:POKE36877,78
:GOTO810
•1000 IFX=HTHENCASH=CASH+OD(X)*BET
:GOTO1020
•1010 CA=CA-BE:IFCA=0THENPRINT"{SC
}YOU LOST ALL YOUR      MONEY-BETT
ER LUCK NEXTTIME!":END
•1020 PRINT"{SC}{CD}{CD}{CD}{CD}{C
D}{CD}YOU NOW HAVE $";CASH
•1030 FORX=1TO5000:NEXT
•1040 CNT=CNT+1:GOTO480
•2020 PRINT "{SC}":POKE36879,88
•2040 PRINT"     M{YL}) {G1} M{BL}£
 {G1} M{PU}£ {G1} M{WH}£"
•2050 PRINT"{G1}    {BK}COMMODORE
DOWNS{G1} "
•2060 PRINT"{CR}{CR}{CR}{CR}{BK}*{
RV}           {RO}£"
•2070 PRINT"{CR}{CR}{CR}{CR}{CR}{W
H}*{RV}{WH} ++++++++ {RO})"
•2080 PRINT"{CR}{CR}{CR}{CR}{CR}{C
R}{WH}{RV} ++++++++ ":PRINT"{CR}{
CR}{CR}{CR}{CR}{CR}{G1}{RV}
```

```
"
•2090 PRINT "{WH}POYYPOYYPOYYPOYYP
OYYPOY"
•2100 PRINT "{CU}{BK}{RV}1{BR}
    {20 SPACES}    {BK}F"
•2110 PRINT "{CU}{BK}{RV} {BR}
    {20 SPACES}    {BK} "
•2120 PRINT "{CU}{BK}{RV}2{BR}
    {20 SPACES}    {BK}I"
•2130 PRINT "{CU}{BK}{RV} {BR}
    {20 SPACES}    {BK} "
•2140 PRINT "{CU}{BK}{RV}3{BR}
    {20 SPACES}    {BK}N"
•2150 PRINT "{CU}{BK}{RV} {BR}
    {20 SPACES}    {BK} "
•2160 PRINT "{CU}{BK}{RV}4{BR}
    {20 SPACES}    {BK}I"
•2170 PRINT "{CU}{BK}{RV} {BR}
    {20 SPACES}    {BK} "
•2180 PRINT "{CU}{BK}{RV}5{BR}
    {20 SPACES}    {BK}S"
•2190 PRINT "{CU}{BK}{RV} {BR}
    {20 SPACES}    {BK} "
•2200 PRINT "{CU}{BK}{RV}6{BR}
    {20 SPACES}    {BK}H"
•2210 PRINT "{CU}{BK}{RV} {BR}
    {20 SPACES}    {BK} "
•2220 PRINT "{WH}{CU}POYYPOYYPOYYP
OYYPOYYPO"
•2230 RETURN
•2300 DATA COLE SLEW,SECRETARY,LIG
HTNIN',SLO POKE,STREAKER,GAMBLER
•2310 DATA FIREWORKS,HOOFER,THUNDE
R,SLICTICK,GOLD BOY,ROCKET
•2320 DATA OLD FORB,AFFIRMED,ASSEM
BLER,DANCERS,HILLY,NASHUA
•2330 DATA GEM RISK,RIVALRY,PENSIV
E,ASSAULT,TIM TAM,CARRY OUT
•2340 DATA FRENCHY,NO HASTE,BREVIT
Y,NASHUA,FABIUS,ASSAULT
•2350 DATA MAJESTY,PENSIVE,WORTH,O
MAHA,JET-SET,COUNT IT
•2360 DATA DAUBER,CROZIER,SHAM,CAR
RY OUT,PONDER,NO LUCK
•2370 DATA ADVOCADO,PRINCE,SPY SON
G,MISSTEP,ZAL,TICKET
•2380 DATA LEG ACHE,CAPOT,CITATION
,NEEDLES,DECIDED,SWAPS
•2390 DATA LIL PASS,RUMBO,BUG KING
,CAVALCADE,DEAD PONY,SLO DANCE
```

BUG REPELLENT LINE CODES FOR POST TIME (VIC 20)

```
LINE # 5:AF
LINE # 10:BA
LINE # 15:LI
LINE # 20:PG
LINE # 25:LK
LINE # 30:OH
LINE # 35:KK
LINE # 40:PM
LINE # 45:FC
LINE # 50:NL
LINE # 55:BI
LINE # 61:GP
LINE # 62:MD
LINE # 63:MG
LINE # 64:LB
LINE # 65:KN
LINE # 66:GH
LINE # 70:HC
LINE # 110:HD
LINE # 120:EJ
LINE # 130:LB
LINE # 140:EJ
LINE # 180:JF
LINE # 190:OG
LINE # 195:AP
LINE # 200:CG
LINE # 205:JC
LINE # 210:NI
LINE # 220:BC
LINE # 230:HD
LINE # 466:HM
LINE # 470:GD
LINE # 480:CD
LINE # 490:DF
LINE # 500:DL
LINE # 510:FN
LINE # 520:IB
LINE # 525:KN
LINE # 530:GC
LINE # 540:NH
LINE # 543:DB
LINE # 550:CA
LINE # 560:CO
LINE # 640:PC
LINE # 650:DJ
LINE # 655:BE
LINE # 660:ND
LINE # 700:FB
LINE # 710:KL
LINE # 750:CG
LINE # 810:BN
LINE # 820:BD
LINE # 830:LB
LINE # 840:FK
LINE # 850:OJ
LINE # 860:MK
LINE # 870:PB
LINE # 880:PC
LINE # 890:NH
LINE # 900:FE
LINE # 1000:KP
LINE # 1010:OL
LINE # 1020:EO
LINE # 1030:AG
LINE # 1040:LE
LINE # 2020:DA
LINE # 2040:NN
LINE # 2050:CG
LINE # 2060:ED
LINE # 2070:PC
LINE # 2080:CE
LINE # 2090:NC
LINE # 2100:KN
LINE # 2110:BL
LINE # 2120:OB
LINE # 2130:BL
LINE # 2140:OD
LINE # 2150:BL
LINE # 2160:NP
LINE # 2170:BL
LINE # 2180:PA
LINE # 2190:BL
LINE # 2200:OG
LINE # 2210:BL
LINE # 2220:EE
LINE # 2230:IM
LINE # 2300:MF
LINE # 2310:II
LINE # 2320:FJ
LINE # 2330:HP
LINE # 2340:DA
LINE # 2350:KC
LINE # 2360:CP
LINE # 2370:OE
LINE # 2380:KN
LINE # 2390:BO
LINES: 96
```

MEMORY MANAGEMENT

FROM PAGE 95

TEST 3

```
10 REM:FOR TAPE-SAVE FIRST
20 REM:MEMORY MANAGEMENT TEST 3
30 REM:LOADS INTO FIRST 8K EXPANDER
40 REM: IN THE VIC-20. WILL NEED THE 3K
50 REM: EXPANDER AND 8K IN BLOCK 5 AS WELL
60 IF PEEK(251)=1 GOTO 100
70 T3$="{SC}TEST 3 NOW ACTIVE":PRINTT3$
80 PRINT"LOADING TEST 4 INTO 3K"
90 POKE 1024,0:POKE44,4:LOAD"TEST 4",8:REM**LEAVE OFF THE ,8 FOR TAPE
100 PRINT T3$
110 PRINT "ENTER {RV}4{RO} FOR TEST 4"
120 PRINT "ENTER {RV}5{RO} FOR TEST 5"
130 GET IN$:IF IN$=""GOTO130
140 IN=VAL(IN$)
150 IF IN<4 OR IN>5GOTO110
160 IF IN=4 THEN POKE 44,4:GOTO10:REM 3K EXPANDER
170 POKE 44,160:GOTO10:REM BLOCK 5
```

TEST 4

```
10 REM:FOR TAPE-SAVE SECOND
20 REM:MEMORY MANAGEMENT TEST 4
30 REM:LOADS INTO 3K EXPANSION AREA
40 REM: IN THE VIC-20. WILL NEED THE 3K
50 REM: EXPANDER AND 8K IN BLOCK 5 AS WELL
60 IF PEEK(251)=1 GOTO 100
70 T4$="{SC}TEST 4 NOW ACTIVE":PRINTT4$
80 PRINT"LOADING TEST 5 INTO   BLOCK 5"
90 POKE 40960,0:POKE44,160:LOAD"TEST 5",8:REM**LEAVE OFF THE ,8 FOR TAPE
100 PRINT T4$
110 PRINT "ENTER {RV}3{RO} FOR TEST 3"
120 PRINT "ENTER {RV}5{RO} FOR TEST 5"
130 GET IN$:IF IN$=""GOTO130
140 IN=VAL(IN$)
150 IF IN<3 OR IN>5GOTO110
155 IF IN=4 GOTO 100
160 IF IN=3 THEN POKE 44,18:GOTO10:REM 8K EXPANDER
170 POKE 44,160:GOTO10:REM BLOCK 5
```

TEST 5

```
10 REM:FOR TAPE-SAVE THIRD
20 REM:MEMORY MANAGEMENT TEST 5
30 REM:LOADS INTO 8K BLOCK 5 IN THE VIC-20
40 REM: THE VIC-20. WILL NEED THE 3K
50 REM: EXPANDER AND 8K IN BLOCK 5 AS WELL
60 IF PEEK(251)=1 GOTO 100
70 T5$="{SC}TEST 5 NOW ACTIVE":PRINTT5$
80 POKE 251,1:REM SET LOAD COMPLETE FLAG
90 GOTO 110
100 PRINT T5$
110 PRINT "ENTER {RV}3{RO} FOR TEST 3"
120 PRINT "ENTER {RV}4{RO} FOR TEST 4"
130 GET IN$:IF IN$=""GOTO130
140 IN=VAL(IN$)
150 IF IN<3 OR IN>4GOTO110
160 IF IN=3 THEN POKE 44,18:GOTO10:REM 8K EXPANDER
170 POKE 44,4:GOTO10:REM 3K EXPANDER
```

TEST 6

```
10 REM:MEMORY MANAGEMENT TEST 6
20 REM:LOADS INTO NORMAL BASIC RAM ON THE C64
30 POKE 44,192:POKE49152,0:LOAD"TEST 7",8
```

TEST 7

```
10 REM:MEMORY MANAGEMENT TEST 7
20 REM:LOADS INTO FREE RAM AT ADDRESS 49152 IN THE C64
30 POKE 45,3:POKE 46,8:CLR
40 PRINT"BYTES FREE ARE"2^16+FRE(0)
50 PRINT"TEST 6 HAS BEEN ELIMINATED"
```

```
60 PRINT"ALL OF ORIGINAL BASIC RA
 M IS AVAILABLE  FOR STORAGE OF DA
 TA"
```

TEST 8

```
10 REM:FOR TAPE-SAVE FIRST
20 REM:MEMORY MANAGEMENT TEST 8
30 REM:LOADS INTO STANDARD RAM IN
 THE   COMMODORE 64
60 IF PEEK(251)=1 GOTO 100
70 T8$="{SC}TEST 8 NOW ACTIVE":PR
 INT T8$
80 PRINT"LOADING TEST 9 AT 49152"
90 POKE 49152,0:POKE44,192:LOAD"T
 EST 9",8:REM**LEAVE OFF THE ,8 FO
 R TAPE
100 PRINT T8$
110 PRINT "ENTER {RV}9{RO} FOR TE
 ST 9"
130 GET IN$:IF IN$=""GOTO130
140 IN=VAL(IN$)
150 IF IN<>9 GOTO 110
170 POKE 44,192:GOTO 10:REM BLOCK
 49152
```

TEST 9

```
10 REM:FOR TAPE-SAVE SECOND
20 REM:MEMORY MANAGEMENT TEST 9
30 REM:LOADS INTO $C000 BLOCK IN
 THE   COMMODORE 64
60 IF PEEK(251)=1 GOTO 100
70 T9$="{SC}TEST 9 NOW ACTIVE":PR
 INTT9$
80 POKE 251,1:REM SET LOAD COMPLE
 TE FLAG
90 GOTO 110
100 PRINT T9$
110 PRINT "ENTER {RV}8{RO} FOR TE
 ST 8"
130 GET IN$:IF IN$=""GOTO130
140 IN=VAL(IN$)
150 IF IN<>8 GOTO 110
170 POKE 44,8:GOTO 10:REM BACK TO
 NORMAL RAM
```

BUG REPELLENT LINE CODES FOR MEMORY MANAGEMENT

TEST 3

```
LINE # 10:LL
LINE # 20:CB
LINE # 30:IM
LINE # 40:PA
LINE # 50:KB
LINE # 60:DH
LINE # 70:OA
LINE # 80:LL
LINE # 90:OO
LINE # 100:EL
LINE # 110:PC
LINE # 120:PA
LINE # 130:DM
LINE # 140:CK
LINE # 150:HF
LINE # 160:KH
LINE # 170:FE
LINES: 17
```

TEST 4

```
LINE # 10:MH
LINE # 20:CA
LINE # 30:FK
LINE # 40:PA
LINE # 50:KB
LINE # 60:DH
LINE # 70:IL
LINE # 80:DA
LINE # 90:PJ
LINE # 100:EI
LINE # 110:AM
LINE # 120:PA
LINE # 130:DM
LINE # 140:CK
LINE # 150:GE
LINE # 155:EK
LINE # 160:PI
LINE # 170:FE
LINES: 18
```

TEST 5

```
LINE # 10:JE
LINE # 20:DP
LINE # 30:KC
LINE # 40:AM
LINE # 50:KB
LINE # 60:DH
LINE # 70:FK
LINE # 80:ND
LINE # 90:CC
LINE # 100:FB
LINE # 110:AM
LINE # 120:PC
LINE # 130:DM
LINE # 140:CK
LINE # 150:HF
LINE # 160:PI
LINE # 170:EP
LINES: 17
```

TEST 6

```
LINE # 10:DO
LINE # 20:EC
LINE # 30:MB
LINES: 3
```

TEST 7

```
LINE # 10:DN
LINE # 20:GC
LINE # 30:PC
LINE # 40:BM
LINE # 50:HF
LINE # 60:PN
LINES: 6
```

TEST 8

```
LINE # 10:LL
LINE # 20:DM
LINE # 30:AM
LINE # 60:DH
LINE # 70:EP
LINE # 80:KG
LINE # 90:FG
LINE # 100:EM
LINE # 110:HI
LINE # 130:DM
LINE # 140:CK
LINE # 150:PI
LINE # 170:BH
LINES: 13
```

TEST 9

```
LINE # 10:MH
LINE # 20:DL
LINE # 30:DJ
LINE # 60:DH
LINE # 70:GO
LINE # 80:ND
LINE # 90:CC
LINE # 100:FF
LINE # 110:HK
LINE # 130:DM
LINE # 140:CK
LINE # 150:OJ
LINE # 170:EL
LINES: 13
```

ALPINER

FROM PAGE 93

```
1 REM *****  A L P I N E R  *****
*
2 REM ** DESIGNED BY-BOB LLORET *
*
3 REM **** FOR AHOY' MAGAZINE ***
*
4 REM ============================
=
10 PRINT "{SC}":POKE 53280,6:POKE
 53281,0
20 PRINT TAB(18);"{GN}THE":PRINT
TAB(11);"£ A L P I N E R £"
25 PRINT TAB(13);"{PU}BY:BOB LLOR
ET"
30 PRINT TAB(6);"{RD}{CD}{CD}{CD}
{CD}NUMBER OF PLAYERS:";:INPUT NP
40 IF NP<1 OR NP>4 THEN 10
50 PRINT SPC(6);"{BR}{CD}{CD}ENTE
R PLAYER'S FIRST NAMES"
55 FOR I=1 TO NP
60 PRINT TAB(6);"{YL}{CD}PLAYER #
";I;:INPUT NAME$(I)
70 NEXT I
98 REM ***  DIFFICULTY LEVEL  ***
99 REM ============================
100 PRINT "{SC}":POKE 53280,0:POK
E 53281,7
110 PRINT TAB(10);"{BR}{RV}ACCCCC
CCCCCCCCCCCCS"
120 PRINT TAB(10);"{RV}B * PLAYER
 LEVEL * B"
130 PRINT TAB(10);"{RV}ZCCCCCCCCC
CCCCCCCCCX"
140 PRINT "{BL}{RV}{CD}{CD}{CD}{C
D}{CD}";TAB(13);"[1]{RO} AMATURE"
:PRINT TAB(13);"{PU}{CD}{RV}[2]{R
O} PROFESSIONAL"
150 PRINT TAB(13);"{RD}{CD}{RV}[3
]{RO} ALPINER":PRINT TAB(12);"{BR
}{CD}{CD}{CD}{CD}{CD}{CD}YOUR CHO
ICE {RV}[1-3]{RO}"
160 GET CH$:IF CH$="" THEN 160
170 IF CH$="1" THEN S=13:OBS=6
175 IF CH$="2" THEN S=12:OBS=5
180 IF CH$="3" THEN S=10:OBS=3
190 PRINT "{SC}":POKE 53280,6:POK
E 53281,3
200 PRINT "{BK}{CD}{CD}{CD}{CD}{C
D}{CD}   HOLD ON...I'M ARRANGING
THE COURSE
210 GOTO 1000
220 GOSUB 1200:GOSUB 900
225 FOR T=1 TO NP
230 PRINT "{SC}":POKE 53280,14:PO
KE 53281,1:X=160:Y=145:SC(T)=5000
0:M(T)=0:GA=0
235 FOR A=1 TO 10:PRINT:NEXT A
240 PRINT TAB(12);"{BL}GET READY,
 ";NAME$(T)
245 FOR A=1 TO 11:PRINT:NEXT A
250 POKE V,X:POKE V+1,Y:POKE V+21
,1
255 FOR D=1 TO 1500:NEXT D
260 SI=54272:FOR I=0 TO 24:POKE S
I+I,0:NEXT I
265 REM ****  MAIN 1ST LOOP  ****
266 REM =====================
270 FOR COUNT=1 TO 600:C=INT(RND(
0)*11)
280 PRINT "{GN}";TAB(C);CHR$(94);
TAB(C+28);CHR$(94)
285 POKESI+24,3:POKESI+5,64:POKES
I+6,130:POKESI+1,17:POKESI,37:POK
ESI+4,129
290 SYS(49400)
300 IF PEEK(V+31)ANDX=XTHENPOKE20
40,14:SC(T)=SC(T)-150:M(T)=M(T)+1
320 B=B+1:IF B>=S THEN C2=INT(RND
(0)*15)+10:GOTO 340
330 GOTO 350
340 PRINT"{BL}{CU}";TAB(C2);CHR$(
95);TAB(C2+5);CHR$(95):B=0:POKE20
40,13:GA(T)=GA(T)+1
350 NEXT COUNT
355 REM ***  END OF RUN  ***
356 REM ====================
360 FOR A=1 TO 30:C=INT(RND(0)*11
):PRINT"{GN}";TAB(C);CHR$(94);TAB
(C+28);CHR$(94)
365 NEXT A
370 POKE SI+4,16:POKE SI+1,0:POKE
 SI,0
380 LE$="FIRST":GOSUB 800
385 REM ****  SECOND LEG  ****
386 REM =====================
390 PRINT "{SC}":POKE 53280,14:PO
KE 53281,1:X=160:Y=145:M=0
400 FOR A=1 TO 10:PRINT:NEXT A
410 PRINT TAB(12);"{BL}GET READY,
 ";NAME$(T)
420 FOR A=1 TO 11:PRINT:NEXT A
430 POKE V,X:POKE V+1,Y:POKE V+21
,1
```

```
•440 FOR D=1 TO 1000:NEXT D
•450 POKESI+24,2:POKESI+5,64:POKES
 I+6,130:POKESI+1,17:POKESI,37:POK
 ESI+4,129
•455 REM **** MAIN 2ND LOOP ****
•456 REM ======================
•460 FOR COUNT=1 TO 700:C=INT(RND(
 0)*11):C1=INT(RND(0)*15)+11
•470 PRINT "{GN}";TAB(C);CHR$(94);
 TAB(C+28);CHR$(94)
•475 IF B=OBS THEN 477
•476 GOTO 480
•477 PRINT TAB(C1);"{G2}";CHR$(96)
•480 SYS(49400)
•490 IF PEEK(V+31)ANDX=X THEN POKE
  2040,14:SC(T)=SC(T)-250:M(T)=M(T
 )+1
•500 B=B+1:IF B>=S THEN C2=INT(RND
 (0)*15)+10:GOTO 520
•510 GOTO 530
•520 PRINT "{BL}{CU}";TAB(C2);CHR$
 (95);TAB(C2+5);CHR$(95):B=0:POKE
 2040,13
•530 NEXT COUNT
•540 FOR A=1 TO 25:C=INT(RND(0)*11
 ):PRINT "{GN}";TAB(C);CHR$(94);TA
 B(C+28);CHR$(94)
•545 NEXT A
•550 POKE SI+4,16:POKE SI+1,0:POKE
  SI,0
•560 LE$="SECOND":GOSUB 800
•565 REM ****  THIRD LEG  ****
•566 REM =====================
•570 PRINT "{SC}":POKE 53280,14:PO
 KE 53281,1:X=160:Y=145:M=0
•575 FOR A=1 TO 10:PRINT:NEXT A
•580 PRINT TAB(12);"{BL}GET READY,
  ";NAME$(T)
•585 FOR A=1 TO 11:PRINT:NEXT A
•590 POKE V,X:POKE V+1,Y:POKE V+21
 ,1
•595 FOR D=1 TO 1000:NEXT D
•600 POKESI+24,2:POKESI+5,64:POKES
 I+6,130:POKESI+1,17:POKESI,37:POK
 ESI+4,129
•605 REM **** MAIN 3RD LOOP ****
•606 REM ======================
•610 FOR COUNT=1 TO 800:C=INT(RND(
 0)*11):C1=INT(RND(0)*15)+9
 620 PRINT "{GN}";TAB(C);CHR$(94);
 TAB(C+28);CHR$(94)
•630 IF B=OBS THEN 650
•640 GOTO 660
•650 PRINT TAB(C1);"{G2}";CHR$(96)
 ;TAB(C1+5);"{BR}{CU}";CHR$(100);C
 HR$(101);CHR$(102)
•660 SYS(49400)
•670 IF PEEK(V+31)ANDX=X THEN POKE
  2040,14:SC(T)=SC(T)-350:M(T)=M(T
 )+1
•680 B=B+1:IF B>=S THEN C2=INT(RND
 (0)*15)+10:GOTO 700
•690 GOTO 710
•700 PRINT "{BL}{CU}";TAB(C2);CHR$
 (95);TAB(C2+5);CHR$(95):B=0:POKE
 2040,13
•710 NEXT COUNT
•720 FOR A=1 TO 25:C=INT(RND(0)*11
 ):PRINT "{GN}";TAB(C);CHR$(94);TA
 B(C+28);CHR$(94)
•730 NEXT A
•740 POKE SI+4,16:POKE SI+1,0:POKE
  SI,0
•750 LE$="THIRD":GOSUB 800
•760 IF LE$="THIRD" AND T=NP THEN
 END
•770 NEXT T
•798 REM ****  RUN RESULTS  ****
•799 REM ======================
•800 PRINT "{SC}":POKE V+21,0:POKE
  53280,0:POKE 53281,6
•810 PRINT "{BK}";TAB(9);"* ";LE$;
 " LEG RESULTS *"
•815 PRINT "{CD}{YL}   ";NAME$(1);
 ":"
•816 PRINT "{WH}";TAB(8);"NO. OF G
 ATES....";GA(1)
•817 PRINT TAB(8);"OBSTICLES HIT..
 .";M(1)
•818 PRINT TAB(8);"TOTAL SCORE....
 .";SC(1)
•820 IF T<2 THEN FOR A=1 TO 16:PRI
 NT:NEXT A:GOTO 860
•825 PRINT "{YL}   ";NAME$(2);":"
•826 PRINT "{WH}";TAB(8);"NO. OF G
 ATES....";GA(2)
•827 PRINT TAB(8);"OBSTICLES HIT..
 .";M(2)
•828 PRINT TAB(8);"TOTAL SCORE....
 .";SC(2)
•830 IF T<3 THEN FOR A=1 TO 12:PRI
 NT:NEXT A:GOTO 860
•835 PRINT "{YL}   ";NAME$(3);":"
•836 PRINT "{WH}";TAB(8);"NO. OF G
 ATES....";GA(3)
•837 PRINT TAB(8);"OBSTICLES HIT..
```

```
.";M(3)
•838 PRINT TAB(8);"TOTAL SCORE....
.";SC(3)
•840 IF T<4 THEN FOR A=1 TO 8:PRIN
T:NEXT A:GOTO 860
•845 PRINT "{YL}  ";NAME$(4);":"
•846 PRINT "{WH}";TAB(8);"NO. OF G
ATES....";GA(4)
•847 PRINT TAB(8);"OBSTICLES HIT..
.";M(4)
•848 PRINT TAB(8);"TOTAL SCORE....
.";SC(4)
•850 FOR A=1 TO 3:PRINT:NEXT A
•860 IF LE$<>"THIRD" THEN MSG$="NE
XT LEG COMING UP"
•870 IF LE$="THIRD" AND T<>NP THEN
 MSG$="NEXT PLAYER PLEASE"
•880 IF LE$="THIRD" AND T=NP THEN
MSG$="* FINAL RESULTS *"
•885 CENT=INT(40-LEN(MSG$))/2
•890 PRINT TAB(CENT);"{YL}";MSG$:F
OR D=1 TO 4500:NEXT D:RETURN
•895 CLR:END
•896 REM ===========================
======
•900 FOR I=1 TO 101:READ A:POKE 49
151+I,A:NEXT I
•910 FOR I=1 TO 19:READ A:POKE 493
99+I,A:NEXT I:RETURN
•1000 REM **** CHAR CHANGE ****
•1005 REM ======================
•1010 PRINT CHR$(142):POKE 52,48:P
OKE 56,48
•1020 POKE 56334,PEEK(56334)AND254
:POKE 1,PEEK(1)AND251
•1030 FOR I=0 TO 511:POKE I+12288,
PEEK(I+53248):NEXT
•1040 POKE 1,PEEK(1)OR4:POKE 56334
,PEEK(56334)OR1:POKE53272,(PEEK(5
3272)AND240)+12
•1050 READ LOC:IF LOC=-1 THEN 1100
•1060 FOR A=0 TO 7:READ CH:POKE LO
C+A,CH:NEXT A:GOTO 1050
•1070 DATA 12528,24,24,60,126,255,
24,24,24
•1080 DATA 12536,112,124,126,127,9
6,96,96,96
•1090 DATA 12512,0,2,252,124,60,12
4,236,196
•1100 DATA 12800,48,124,126,62,62,
127,255,255
•1110 DATA 12832,12,3,127,191,191,
127,0,0
•1120 DATA 12840,0,0,255,255,255,2
55,1,6
•1130 DATA 12848,0,0,252,254,254,2
52,128,0,-1
•1150 GOTO 220
•1198 REM *** DATA FOR SPRITES ***
•1199 REM ========================
•1200 V=53248:POKE 2040,13:POKE V+
39,6:POKE V+28,3:POKE V+38,10:POK
E V+37,9
•1210 FORA=0TO 62:READ CH:POKE 832
+A,CH:NEXT A
•1220 DATA 0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,190,0,0,190,0,0,255,0,15
•1230 DATA 255,240,51,195,204,0,25
5,0,0,255,0,0,130,0,0,130
•1240 DATA 0,0,130,0,0,130,0,0,130
,0,0,0,0,0,0,0,0,0,0,0,0,0
•1250 POKE 2041,14:POKE V+40,6:POK
E V+38,10:POKE V+37,9
•1255 FORA=0TO 62:READ CH:POKE 896
+A,CH:NEXT A
•1260 DATA 0,0,0,0,0,0,0,0,0,0,192
,192,8,51,192,8,15,192,8,12,192
•1270 DATA 8,63,0,11,255,192,8,12,
48,8,12,0,8,12,0,8,8,0,0,8,0,0,8,
0
•1280 DATA 0,8,0,0,8,0,0,0,0,0,0,0
,0,0,0,0,0,0,0
•1300 RETURN
•1400 DATA 173,0,220,74,176,3,206,
1,208,74,176,3,238,1,208,74,176,3
8,173
•1410 DATA 0,208,208,15,173,16,208
,41,1,240,12,173,16,208,41,254,14
1,16
•1420 DATA 208,206,0,208,96,173,16
,208,9,1,162,63,141,16,208,142,0,
208,96
•1430 DATA 74,176,32,238,0,208,240
,28,173,16,208,41,1,240,20,169,64
,205
•1440 DATA 0,208,208,13,173,16,208
,41,254,162,0,141,16,208,142,0,20
8,96
•1450 DATA 173,16,208,9,1,141,16,2
08,96
•1460 DATA 32,0,192,32,0,192,32,0,
192,32,0,192,96,32,0,192,76,5,193
```

BUG REPELLENT LINE CODES FOR ALPINER

```
LINE # 1:AL
LINE # 2:BF
LINE # 3:HK
LINE # 4:EL
LINE # 10:OJ
LINE # 20:CI
LINE # 25:CN
LINE # 30:MA
LINE # 40:FI
LINE # 50:KP
LINE # 55:PA
LINE # 60:PM
LINE # 70:MN
LINE # 98:PA
LINE # 99:ME
LINE # 100:OI
LINE # 110:EH
LINE # 120:PI
LINE # 130:GO
LINE # 140:MD
LINE # 150:JF
LINE # 160:BJ
LINE # 170:IO
LINE # 175:GF
LINE # 180:KM
LINE # 190:OK
LINE # 200:PI
LINE # 210:GN
LINE # 220:PB
LINE # 225:GH
LINE # 230:JM
LINE # 235:BD
LINE # 240:PH
LINE # 245:AM
LINE # 250:AM
LINE # 255:IL
LINE # 260:KB
LINE # 265:GA
LINE # 266:OO
LINE # 270:DC
LINE # 280:EA
LINE # 285:MJ
LINE # 290:PF
LINE # 300:DD
LINE # 320:OF
LINE # 330:EC
LINE # 340:AO
LINE # 350:DN
LINE # 355:JD
LINE # 356:LE
LINE # 360:JD
LINE # 365:OF
LINE # 370:FI
LINE # 380:OD
LINE # 385:DG
LINE # 386:DF
LINE # 390:BJ
LINE # 400:BD
LINE # 410:PH
LINE # 420:AM
LINE # 430:AM
LINE # 440:JO
LINE # 450:PK
LINE # 455:LP
LINE # 456:GJ
LINE # 460:FA
LINE # 470:EA
LINE # 475:BP
LINE # 476:EA
LINE # 477:JM
LINE # 480:PF
LINE # 490:FN
LINE # 500:CB
LINE # 510:EG
LINE # 520:KN
LINE # 530:DN
LINE # 540:PG
LINE # 545:OF
LINE # 550:FI
LINE # 560:NL
LINE # 565:BL
LINE # 566:OO
LINE # 570:BJ
LINE # 575:BD
LINE # 580:PH
LINE # 585:AM
LINE # 590:AM
LINE # 595:JO
LINE # 600:PK
LINE # 605:HK
LINE # 606:GJ
LINE # 610:LP
LINE # 620:EA
LINE # 630:BC
LINE # 640:EM
LINE # 650:ND
LINE # 660:PF
LINE # 670:PO
LINE # 680:CF
LINE # 690:EC
LINE # 700:KN
LINE # 710:DN
LINE # 720:PG
LINE # 730:OF
LINE # 740:FI
LINE # 750:LK
LINE # 760:OP
LINE # 770:PC
LINE # 798:BO
LINE # 799:GJ
LINE # 800:KF
LINE # 810:KD
LINE # 815:EI
LINE # 816:EO
LINE # 817:NP
LINE # 818:OM
LINE # 820:DC
LINE # 825:BA
LINE # 826:EP
LINE # 827:NE
LINE # 828:PD
LINE # 830:HL
LINE # 835:GN
LINE # 836:EM
LINE # 837:NF
LINE # 838:PC
LINE # 840:PH
LINE # 845:GK
LINE # 846:EN
LINE # 847:NC
LINE # 848:OJ
LINE # 850:ML
LINE # 860:CE
LINE # 870:AK
LINE # 880:IA
LINE # 885:OO
LINE # 890:OD
LINE # 895:FB
LINE # 896:FB
LINE # 900:BM
LINE # 910:NI
LINE # 1000:BC
LINE # 1005:OO
LINE # 1010:EM
LINE # 1020:HD
LINE # 1030:KA
LINE # 1040:NO
LINE # 1050:HP
LINE # 1060:OG
LINE # 1070:NJ
LINE # 1080:OG
LINE # 1090:JL
LINE # 1100:LA
LINE # 1110:EB
LINE # 1120:PL
LINE # 1130:DF
LINE # 1150:EE
LINE # 1198:GF
LINE # 1199:LA
LINE # 1200:CI
LINE # 1210:JE
LINE # 1220:HM
LINE # 1230:ED
LINE # 1240:PB
LINE # 1250:IH
LINE # 1255:LO
LINE # 1260:II
LINE # 1270:OF
LINE # 1280:HO
LINE # 1300:IM
LINE # 1400:FC
LINE # 1410:JG
LINE # 1420:JH
LINE # 1430:EK
LINE # 1440:PE
LINE # 1450:JG
LINE # 1460:CP
LINES: 177
```

CORRECTION TO RUPERT REPORT—

HURRAY FOR ARRAYS (MAY *AHOY!*)

The two programs listed on the following page were omitted from May's *Rupert Report—Hurray for Arrays!* They are referred to in the article as Listing 1 and Listing 2. We apologize for the error, and regret the necessity to print the listings in reduced size. Space, unfortunately, is at too much of a premium this issue to run the programs full-size.

LISTING 1

```
10 DIM N(39)
20 GOSUB 2000:REM STORE CURSOR ROUTINE
30 GOSUB 3000:REM INITIALIZE SOUND
40 BALL$=CHR$(113)
50 PRINT CHR$(147):REM CLEAR SCREEN
60 ROW=21:COL=0:GOSUB 1000
70 PRINT"---------------------------------------"
85 REM ********************************
90 REM ******START OF MAIN LOOP*******
95 REM ********************************
100 COL=20:ROW=0
110 FOR ROW=0 TO 20
115 REM  CHOOSE DIRECTION TO BE 1 OR -1
120 DIR=INT(RND(1)*2)
130 IF DIR=0 THEN DIR=-1
140 COL=COL+DIR:GOSUB 1000:REM POSITION    CURSOR AT ROW,COL
150 PRINT BALL$;
160 REM - IF BELOW STACK OF BALLS, DON'T    PRINT DOT
170 IF ROW >= 25-N(COL) THEN 190
175 REM - BACK UP AND REPLACE BALL WITH    DOT
180 PRINT CHR$(157)".";
190 NEXT ROW
200 N(COL)=N(COL)+1
210 ROW=25-N(COL):GOSUB 1000
220 PRINT BALL$;
230 GOSUB 3100  :REM KERPLUNK SOUND
240 GOTO 100
245 REM *******************************
250 REM ******END OF MAIN LOOP********
255 REM *******************************
900 END
995 REM================================
996 REM--ROUTINE TO POSITION THE CURSOR
997 REM================================
998 REM - CALL IT AFTER DEFINING ROW,COL    WHERE ROW=0 TO 24, COL=0 TO 39
1000 IF ROW<0 OR ROW>24 THEN 1040
1010 IF COL<0 OR COL>39 THEN 1040
1020 POKE SA+1,ROW:POKE SA+3,COL
1030 SYS SA
1040 RETURN
1995 REM================================
1996 REM-PUT CURSOR ROUTINE INTO MEMORY 1997 REM===============================
2000 SA=49152:X=0
2010 READ B:IF B=-1 THEN 2030
2020 POKE SA+X,B:X=X+1:GOTO 2010
2030 RETURN
2040 DATA  162,0,160,0,24
2050 DATA 32,240,255,96,-1
2995 REM================================
2996 REM--INITIALIZE SOUND--        2997 REM===============================
3000 S=54272
3010 FOR L=0 TO 24:POKE S+L,0:NEXT
3020 POKE S+5,6   :REM V1 ATTACK/DECAY
3030 POKE S+6,3,  :REM SUSTAIN/RELEASE
3040 POKE S+15,80 :REM V3 HIGH FREQ
3050 POKE S+24,15 :REM VOLUME
3060 RETURN
3095 REM================================
3096 REM--- RING MODULATION KERPLUNK -  3097 REM===============================
3100 POKE S+1,200*RND(1) :REM V1 FREQ
3110 POKE S+4,21 :REM TRIANGLE,RING MOD
3120 FOR PAUSE=1 TO 100:NEXT
3130 POKE S+4,20 :REM STOP SOUND
3140 RETURN
```

BUG REPELLENT LINE CODES FOR LISTING 1

LINE # 10:JD	LINE # 100:CL	LINE # 180:BC	LINE # 2010:MG
LINE # 20:PJ	LINE # 110:FH	LINE # 190:HN	LINE # 2020:MP
LINE # 30:OA	LINE # 115:GL	LINE # 200:II	LINE # 2030:IM
LINE # 40:LD	LINE # 120:LK	LINE # 210:JO	LINE # 2040:DG
LINE # 50:EG	LINE # 130:GO	LINE # 220:CO	LINE # 2050:LJ
LINE # 60:HF	LINE # 140:DM	LINE # 230:CA	LINE # 2995:BD
LINE # 70:KA	LINE # 150:CO	LINE # 240:CF	LINE # 2996:EA
LINE # 85:CE	LINE # 160:NM	LINE # 245:GH	LINE # 3000:BI
LINE # 90:JL	LINE # 170:ND	LINE # 250:JM	LINE # 3010:BE
LINE # 95:CE	LINE # 175:ON	LINE # 255:GH	LINE # 3020:JC
		LINE # 900:IC	LINE # 3030:CA
		LINE # 995:HD	LINE # 3040:FN
		LINE # 996:OA	LINE # 3050:AG
		LINE # 997:HD	LINE # 3060:IM
		LINE # 998:DD	LINE # 3095:PE
		LINE # 1000:JD	LINE # 3096:JB
		LINE # 1010:EB	LINE # 3100:AK
		LINE # 1020:OH	LINE # 3110:AC
		LINE # 1030:DF	LINE # 3120:PB
		LINE # 1040:IM	LINE # 3130:PC
		LINE # 1995:BD	LINE # 3140:IM
		LINE # 1996:FH	LINES: 64
		LINE # 2000:LM	

LISTING 2

```
10 PRINT CHR$(147)
20 DIM CARD(52),S$(3),V$(13)
30 DEF FNSUIT(N)=INT((CARD(N)-1)/13)
40 DEF FNVLU(N)=CARD(N)-13*FNSUIT(N)
50 GOSUB 100 :REM INITIALIZATION
60 FOR N=1 TO 52:CARD(N)=N:NEXT
70 GOSUB 200 :REM SHOW UNSHUFFLED CARDS
80 GOSUB 300 :REM SHUFFLE CARDS
90 PRINT:GOSUB 200  :REM DISPLAY CARDS
95 END
96 REM==================================
97 REM-DEFINE SUITS AND VALUES SYMBOLS
98 REM==================================
99 REM DIAMONDS, CLUBS, HEARTS, SPADES
100 S$(0)=CHR$(122):S$(1)=CHR$(120)
110 S$(2)=CHR$(115):S$(3)=CHR$(97)
120 FOR N=1 TO 13:READ V$(N):NEXT
130 DATA A,2,3,4,5,6,7,8,9,10,J,Q,K
140 RETURN
195 REM=================================
196 REM---DISPLAY ALL CARDS---
197 REM=================================
200 FOR N=1 TO 52
210 PRINT V$(FNVLU(N));S$(FNSUIT(N))" =";CARD(N),
220 NEXT
230 RETURN
296 REM=================================
297 REM-----SHUFFLE-----
298 REM=================================
300 FOR N=1 TO 52
305 REM - GET CARD TO EXCHANGE
310 E=INT(RND(1)*52)+1
320 T=CARD(N):CARD(N)=CARD(E):CARD(E)=T
330 NEXT
340 RETURN
495 REM================================
496 REM ---DISPLAY CHOSEN CARD---
497 REM================================
500 INPUT "WHICH CARD FROM THE TOP";N
510 PRINT "THE CARD VALUE IS";CARD(N)
520 PRINT "THAT CARD IS ";V$(FNVLU(N));     S$(FNSUIT(N))
```

BUG REPELLENT LINE CODES FOR LISTING 2

LINE # 10:FG	LINE # 197:HD
LINE # 20:NA	LINE # 200:MK
LINE # 30:LL	LINE # 210:LJ
LINE # 40:JH	LINE # 220:IA
LINE # 50:PM	LINE # 230:IM
LINE # 60:FB	LINE # 296:HD
LINE # 70:HC	LINE # 297:HA
LINE # 80:ED	LINE # 298:HD
LINE # 90:FN	LINE # 300:MK
LINE # 95:IC	LINE # 305:GH
LINE # 96:IO	LINE # 310:BE
LINE # 97:IJ	LINE # 320:OH
LINE # 98:IO	LINE # 330:IA
LINE # 99:LE	LINE # 340:IM
LINE # 100:BL	LINE # 495:BD
LINE # 110:ON	LINE # 496:KE
LINE # 120:EH	LINE # 497:BD
LINE # 130:KO	LINE # 500:LK
LINE # 140:IM	LINE # 510:CP
LINE # 195:HD	LINE # 520:MG
LINE # 196:ED	LINES: 41

FLOTSAM

When I first saw *Ahoy!* on the newsstand, I thought it was thin compared to some of your competitors. Then I opened it up and saw all the helpful and interesting articles. Being a typical housewife, I wanted—and got—a lot for my money. As a matter of fact, I was up till 1 a.m. reading it. Both my husband and son enjoyed the magazine as well.

I'm sure that you will grow in pages with age, just like we're growing with our 64. I have full confidence in you.

By the way, I missed your first two issues. Is there any way I can get them, or are they gone forever?

Mrs. Leroy Lilla
Milwaukee, WI

Back issues are now available—see the advertisement on page 97. As far as our page count is concerned, we have definite plans to increase it. We've had doubts in the past about our ability to add pages and retain the level of quality that won us so many readers in the first place. But we finally have the personnel to do it (count the names added to our credit box since our first issue!).

By the way, have you ever counted the number of ads *in those other Commodore books?* Ahoy! *provides a far more favorable ratio of editorial pages to ad pages than any of them. In many cases, that means a higher* total *number of pages of articles, programs, and columns. In many cases, it doesn't. But our quality is always there—and that, judging from what you and readers from all over the U.S. and Canada have told us, is what's most important. (Size, of course, is important too—and we plan to make* Ahoy! *as big and fat as Mike Schneider will let us!)*

Continued on page 93

REVIEWS

Continued from page 58

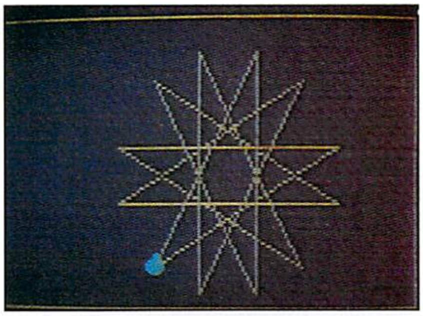
with manuals on BASIC intending to become experts. The only thing we became was frustrated. The material certainly is not that hard; the vocabulary is tiny compared to learning a foreign language. (And most computer languages are pretty foreign!) But the tutorial materials, until very recently, have been the pits.

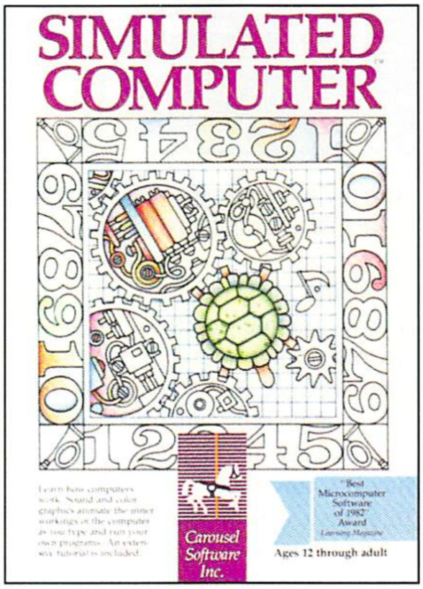
If you persevere, you find that BASIC is not the answer to many real problems and you face the next hurdle—assembly language. Tutorials on assembly language are almost universally arcane tomes which can turn off even the most diehard computer freak. Finally, however, Carousel Software offers some hope for the soon-to-be assembly language programmer.

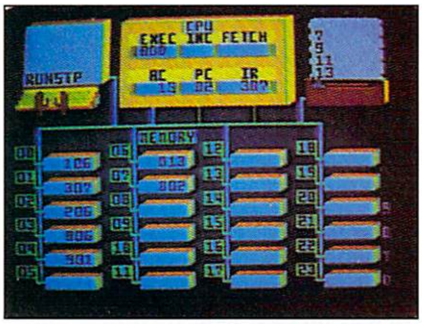
Simulated Computer II gives you a simplified diagram of the logic circuits of a computer and is a painless (although tiny) introduction to assembly language. With it you can create simple turtle graphics, monotone sounds, and short text. By writing brief pseudo-assembly language programs, you can watch the instructions move one-at-a-time through the Simulated Computer and gain a better understanding of what really goes on in those little black chips.

On screen, you see a tiny keyboard with 4 X 8 screen above it, a little printer with a 4 X 7 piece of pin-feed paper sticking up, a CPU (central processing unit), and twenty-four numbered memory locations. The CPU is divided into six boxes which show the command currently being executed, the memory location or register being used, and the number you are working on. They also show the contents of the accumulator, instruction register, and program counter—all locations in the 6510 CPU through which all of the C-64's processing tasks move.

As you type in instructions, a pair of hands types them in on







READER SERVICE NO. 244

the Simulated Computer's keyboard. Each command or number is carried to the appropriate memory location by a visible electric signal which zips along the circuits. Type "RUN" and you will watch that signal carry instructions to the CPU where they are performed one step at a time. Just like a real computer, only about 1000 times slower.

After several instructions, you may hear a note played, watch the little printer clack out a message, or see (momentarily) a full turtle graphics screen while the turtle carries out one task. Four of the Simulated Computer's memory locations are dedicated to special functions—color, turn, and line length for the turtle, and sound. Run the turtle into the edge of the screen and he will crash to a stop.

This is a great representation (although very simplified) of what is really happening in your C-64 and is a good way for a beginner to get a comfortable introduction to assembly language programming. The thirty-five page manual covers the program well, gives sample programs so you can see results right away, and provides you with ten programming problems (and suggested answers).

It does not, however, teach you assembly language; it is just an introduction. *Simulated Computer II* gives you over two dozen instructions to learn. Some, like RUN and LOAD, will be familiar. Only eleven of the instructions are actually used to program the computer and just three of those are real 6510 assembly language instructions. Several others do what could only be accomplished by subroutines of actual assembly language instructions. As an assembly language tutor it does not take you far, but as an aid to becoming comfortable with CPU registers, fetches, increments, and single-mindedness of your computer, it is great.

Carousel Software, Inc., 877 Beacon Street, Boston, MA 02215.

—Richard Herring

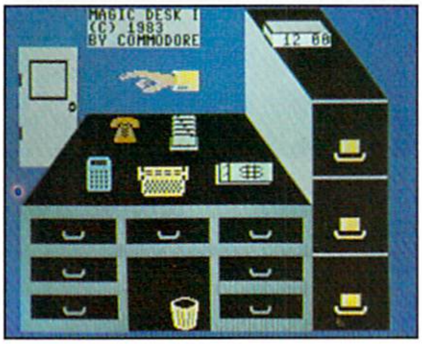
MAGIC DESK I

Commodore Software
C-64
Cartridge

All sorts of software promise "magic" these days—there are Magic-calcs, Magic Paints. . .everything but a Magic Dust Cover. *Magic Desk* is one such program that makes good on its name, noticeably smoothing the transition from working with a typewriter and filing cabinet to controlling their silicon counterparts. "Lisa-like" icons are central to the program's design, and joystick-cursor control adds the convenience of mouse-activated menus that make *Magic Desk* easy to learn and to operate.

Loading's a matter of popping the cart into the game slot and powering up. A hi-res color desk instantly appears, surrounded with the usual office paraphernalia: trash can, file cabinet, phone, clock. The first thing to do is set the digital clock that's on the wall. With the joystick, move the hand-shaped cursor over the clock and hit the fire button to activate the clock—all icons and their corresponding operations are selected in this manner. Punch a few numbers on the keyboard to set the clock, which can then be consulted at any time by returning to this graphic menu. (Too bad it doesn't have a built-in alarm, to make sure you don't get so engrossed in your work that you miss the latest episode of *The A-Team.)*

When the hand hovers over the typewriter, mash the button to move into that mode. Instead of a conventional word processor full of embedded format commands that are harder to decipher than Egyptian hieroglyphics, *Magic Desk* presents a picture-perfect typewriter for you to work with. It's like looking down at the carriage of a standard manual machine, whose realism is backed up with hollow tapping sounds unless you turn them off. With the joystick, you can scroll in all four directions to view the entire document, always the size of an 8½x11″ piece of typing paper. A trackball provides more comfortable and precise cursor control.



***Magic Desk:* a top drawer program.**
READER SERVICE NO. 238

During all stages of operation, icons—graphic symbols—at the bottom depict your options. To print a page, you "mouse over" to the printer icon and hit the button. A page is quickly deleted by literally throwing it into a cartoon-like trash can on the floor. Rarely needed but convenient, a visual "help" screen can be consulted at any time by pressing the Commodore key.

The animated typewriter's black typeface looks sharp and clear while keying in text, but don't expect the capabilities of a word processor. None of the usual features like search and replace, or the ability to move blocks of text, are implemented. And when you delete the left side of a line, the words on the right don't slide over automatically to fill in the space —the deleted section becomes blank space. What this "electronic typewriter" lacks in text-formatting capabilities, it more than makes up for in its ability to directly interface with the filing system. This integrated approach, with both programs running simultaneously, means you can file, retrieve, update, and refile information with more flexibility and less work than with a dedicated database.

Most databases require you to fill out some standard "form," specifying the number of characters per field in the database—which never seems to be as much as you eventually need. Here you just type up a sheet, then file it away in a most familiar manner. Use the hand-cursor (or simply hit "restore") to return to the desk screen. Then position the hand over one of the file cabinet's three drawers. Tap the fire button and a hi-res picture of the disk drive fills the screen, complete with red light to indicate it's in use. Then ten yellow file folders fan out across the screen. Move the cursor to select one, then decide which of that folder's ten pages you want to file this document on. Each folder can be labeled from the keyboard. With three drawers like this, *Magic Desk* provides storage space for up to 300 pages, which may be organized and structured to fit your individual purposes.

Forced to keep tabs on a growing collection of computer humor, I've resorted to filing new jokes and material on a disk devoted to *Magic Desk*: one folder labeled "byte puns," another for "How many teenage girls does it take to boot a disk?" jokes, and a small file for Mexican computer gags (like, what kind of beer do Mexican programmers drink? DOS Equis). Because this kind of material varies in length and content, it's easier to file with *Magic Desk* than with a typical database. Many types of home data, from recipes to financial and auto re-

cords, can be handled efficiently with this filing system.

Magic Desk stores data in relative files, which can't be read by a sequentially based word processor and edited or reformatted more precisely. Relative files also seem to take longer to load and save. Even so, it can streamline many writing and filing operations that don't demand heavy-duty capabilities. And more than a few people will never need anything more than *Magic Desk*. You do need a disk drive, though. (*Magic Desk II* and future upgrades will activate other fixtures on the desk, like the pocket calculator, memo pad, and telephone.)

—Shay Addams

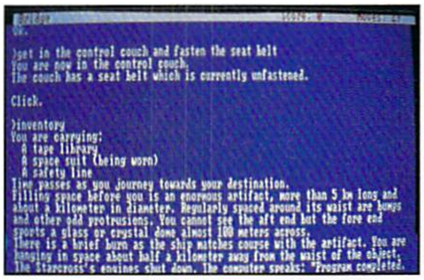
STARCROSS
Infocom
C-64
Disk; keyboard

As Infocom's first science fiction (not fantasy) adventure, *Starcross* is a winner. If you're a sci-fi fan, and not totally addicted to arcade games, you'll enjoy this adventure.

Now, here come the gruesome details. Wait. Stop. One warning. *Starcross* comes with instructions which tell you very little about the game. How it's played, sure. But not much about what you need to do or why. Finding out is part of the game. Read on at your own risk. If you're an inveterate adventurer who would never accept a hint, stop here. I won't give away any of the puzzles, but I may say more than you want to know going in.

Starcross is set about 100 years in the future when the technology exists to harness the power of black holes. Fearless adventurer that you are, you search the solar system in your one-man (or one-woman, if you prefer) ship—the M.C.S. Starcross. As the game begins, you find an uncharted mass. A black hole? Your fortune found? No, a gargantuan ship, drifting in space. Pulled inexorably to one of the docking bays, your ship becomes useless to you. Nothing to do but gain control of the big ship.

No easy matter, this business of getting control. Many systems on the ship are damaged or not working. Your quest will revolve around finding colored crystal rods and placing them in the proper control slots. And wonder of adventure game wonders, the colors make sense. If the critters who escaped from the onboard zoo are killing you every time you go into a dark area of the ship, try the yellow rod in the yellow slot for a little light. I know, I said I wouldn't give anything away, but there are twelve rods, so you'll still have plenty to do.



Your instructions must be earned.
READER SERVICE NO. 239



It is Infocom's parser that reads what you type and turns it into something logical for the computer. Most games feature a 600-word vocabulary.

Some of the best puzzles in *Starcross* are the hiding places for the rods. Don't overlook any nook or cranny where a rod could be tucked away. You'll also find essential stuff (is that vague enough for you?) at each of the big ship's docking ports—the one your ship got pulled into and three others.

The ship itself is actually a tube within a tube, spinning to create an artificial gravity. In the outer tube you will explore over forty different rooms or passageways. Be careful not to miss any. The program will usually tell you about exits up and down, but it won't necessarily tell you that you can go port from, say, the melted spot in the blue hall.

The inner (smaller) tube of the great ship contains grassland and

forest. Remember that gravity will decrease as you approach the axis of the ship's rotation. If you can reach this "no gravity" zone, you'll find a few places which are inaccessible by walking, jumping, or climbing.

Starcross presents some of the best puzzles I've ever seen in an adventure game. In the laboratory, you'll find a blue crystal rod sticking out of a silvery globe. Right there in the lab is nearly everything you need to get that rod. The one item not in the lab is something you should be carrying (or wearing. Sorry, subtlety isn't my forte). Yet, getting the $**!! blue rod is so maddening that. . .that. . .that once you've got it, you'll be glad you bought this game. And, lest I forget, the entrance at the red docking port (one of the first puzzles) is destined to be a classic—along with the Troll Room in the original *Adventure* and the Bank of Zork in the *Wizzard of Frobozz.*

Probably the best thing about *Starcross* (and other Infocom games) is that it's fair. If you've been adventuring a few times, you know that fairness is not a feature found in many games. When I decide to try to kill the spider rather than make friends with him, I know there's a chance I'll buy the farm. (So, I'll have saved the game on a scratch disk before I try. Who wants to replay the whole game?)

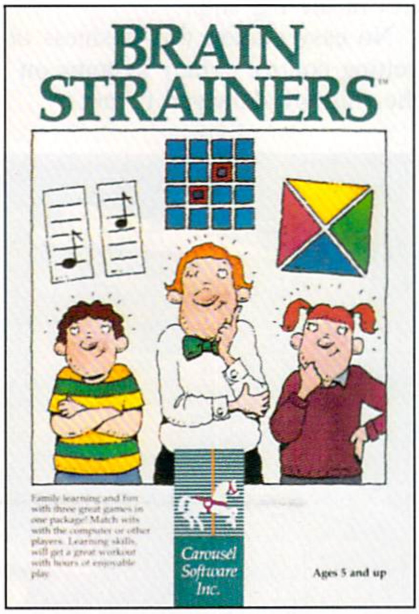
But, when I'm bumbling along through twisty tiny little passages and teeny little twisty passages in some maze, sufficiently lost that any boy scout or girl scout with compass or wet finger would laugh out loud, I don't expect to read, "You're dead. The odorless poison gas in the maze has ended your branch of the family tree." None of that in *Starcross.* All the puzzles are solvable. All the objects are findable. And all the clues are there on the screen (or in your mind's eye or ear).

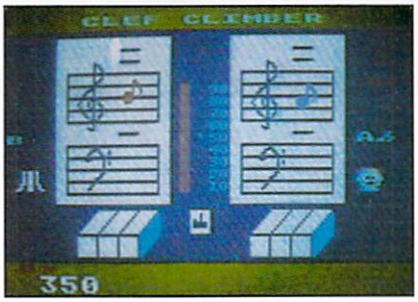
I've played many adventures (two dozen or so) and can heartily recommend *Starcross.* Where else will you find a maintenance mouse "buzzing back and forth looking for garbage, diligently polishing the floor, and waving its 'ears' (small dish antennae) about." I expected the Doctor and K-9 to drop in for a visit at any moment.

To win *Starcross,* you'll need a score of 400 points—achieved by gaining control of the big ship. Lesser scores may earn you the rank of cadet, lieutenant, or planetary commodore. Infocom produces a variety of hint books and maps for its games. If you have a specific question or get hung up at some point, though, drop me a line (P.O. Box 1544, Tallahassee, FL 32302). I'll be glad to offer a clue, or if your note is tear-stained and your control maintained only by antipsychotic drugs, I'll supply a straight answer.

Infocom, Inc., 55 Wheeler St., Cambridge, MA 02138.

—Richard Herring





Includes Clef Climber, Finder's Keepers, Follow the Leader. READER SERVICE NO. 249

BRAIN STRAINERS

Carousel Software
C-64
Disk, cassette; joystick

Brain Strainers is a strain all right, but more on patience and interest than grey matter. And boy, was I fooled! When I started playing the first game called Clef Climber I was hooked by the artful and amusing graphics. Treble clefs, bass clefs, sharps and flats appeared on the monitor—the whole musical ball of wax! "Here's a goodie!" I thought. Wrong! Oh, so wrong.

The object of Clef Climber is to match the musical notes made by the computer. You do this by pushing the joystick forward or backward (translated musically, higher or lower on the scale). One sequence has different notes playing simultaneously, the second has the notes alternating, while another plays the computer's note briefly; then it disappears and you must try to imitate it. A bit confusing? I thought so too, so I turned to the pamphlet accompanying this program only to discover that it must have been translated directly from Japanese, and poorly at that!

What is the point of Clef Climber? I'm not sure. Perhaps a virtuoso would get his jollies trying

to match musical sounds made by a computer, but for most of us tin ears Clef Climber can just keep climbing right off the scale into the great concerto in the sky.

Finder's Keepers is the second game in *Brain Strainers*, and it's a spin-back to that old favorite, *Concentration*. Remember the card game where you spread out a deck face down and tried to match pairs? You got it, Finder's Keepers! It can be played by up to four players at five increasingly difficult levels. Compared to Clef Climber it's sheer delight, but it's not flaw-free. The graphics are minute and sometimes indecipherable, there's no music, and frankly I prefer a deck of cards. But for a rainy day when the kids are starting to climb the walls and you've exhausted all other resources and games, it's fine.

What's really *not* fine is Follow the Leader. In fact, this third game left me absolutely speechless. (Other than "idiotic," "pointless," and "ridiculous" for starters!) It's sort of a ripoff of *Simon*, the annoying game that was popular several Christmases ago.

As in Clef Climber, the object is to match the note the computer makes. Here, however, the computer will increase the number of notes in the pattern when you do it correctly. Pattern? Pattern??? Believe me, after 20 minutes I never got a pattern! So back to the instructional handbook I went, only to discover what the problem was. It said: "Don't play the wrong player's notes or you *loose*." So much for quality control! Believe me, something surely was *loose* when this program was devised.

Brain Strainers is not worth your money, and, most importantly, not worth your kids' time.

Carousel Software, Inc., 877 Beacon Street, Boston, MA 02215.

—Valerie B. Tamis

EDUCATIONAL SOFTWARE

Continued from page 46

BECi, actually decreases the time allowed for responses if correct ones are given. Matching the timing with your child's skill level enhances learning. Arcade games provide a great example. They gradually become harder and faster, constantly challenging you to try to do better. Tom Snyder believes that even a game like *Pac-Man* could be educational if the action could be frozen. The player could be forced to estimate speeds, calculate distances, and analyze escape routes.

In multi-player games, pauses in the action allow for collaboration. Kids are masters at learning and growing as they explore issues with other kids. Do not devalue the learning that takes place during kids' interpersonal interactions. Here is a real strength of your computer. It can provide a group of children with a focal activity, a goal, a challenge, where kids are not only rewarded for the correct answer but for reaching it through *group* participation the first time.

Next month, more ways to judge the educational value of a program. After that, answers to some of the criticisms of educational software and a handy-dandy summary of these articles. Caveat: This series is not intended to persuade you that your child will surely grow up to be a skid row bum if he's not computer literate by the age of nine. Computer company advertising (guilt-tripping) does all the persuading any one person should have to put up with. Computer software is one of many tools which, along with your compassion and interest, can turn your child on to the world in which he'll live and make his own decisions.

Programs for Commodore 64 computers mentioned in this article:

ADD/SUB, Boston Educational Computing Inc., 78 Dartmouth Street, Boston, MA 02116 (617-536-5116). $16.95 on tape, for lower elementary grades.

NUMBER-BECi, Boston Educational Computing Inc., $16.95 on tape, for preschool children.

The Game Show, Advanced Ideas, 1442A Walnut Street, Suite 341, Berkeley, CA 94709 (415-526-9100). $39.95 on disk.

Preschool IQ Builder I, Program Design, Inc., 95 East Putnam Ave., Greenwich, CT 06830 (203-661-8799). $29.95 on disk, $18.95 on tape, for three to six year olds.

Snooper Troops Case #1: The Granite Point Ghost, Spinnaker Software Corporation, 215 First Street, Cambridge, MA 02142 (617-868-4700). $44.95 on disk, for ages ten and up.

Snooper Troops Case #2: The Disappearing Dolphin, Spinnaker Software Corporation, $44.95 on disk, for ages ten and up.

Alphabet Zoo, Spinnaker Software Corporation. $34.95 on cartridge, $29.95 on disk, for prekindergarten to third grade.

Story Machine, Spinnaker Software Corp., $34.95 on disk, for children five to nine years old.

Hey Diddle Diddle, Spinnaker Software Corporation, $29.95 on disk, for children ages three to ten.

Typing Strategy, Behavioral Engineering, 230 Mt. Hermon Road, Scotts Valley, CA 95066. (408-438-5649). $34.95 on cartridge or disk, for all ages.

Typing Tutor/Word Invaders, Academy Software, P.O. Box 9403, San Rafael, CA 94912 (415-499-0850). $24.95 on tape or disk, for all ages.

Eduware Services, Box 22222, 28035 Dorothy Drive, Agoura Hills, CA 91301.

PROGRAM GENERATORS

Continued from page 26

will hold all of our important data.

Here, *TLO* simply asks for "FIELD NAME", "FIELD TYPE" (i.e., is it an ALPHA, a NUMERIC, or a DATE field) and finally the SIZE, in characters, of each field. The end product looks like this:

Count	Field Name	A/N/D	Size
1	Name	Alpha	25
2	Company	Alpha	25
3	Address	Alpha	20
4	A2	Alpha	20
5	City	Alpha	12
6	Zip Code	Alpha	12
7	Telephone #	Alpha	12
8	Comments	Alpha	36

From time to time *TLO* also gives us the chance to review what we have done up to that point and, if necessary, to go back and correct mistakes.

The next major section of the *TLO* is the Flowchart Creation Menu (FCM for short) section:

Flowchart for <PROGRAMNAME>

List flowchart	<1>	Special Functions	<10>
Modify flowchart	<2>	Clear fields	<11>
Code program	<3>	Set file pointer	<12>
Merge flowchart	<4>	Read a file	<14>
Abort	<5>	Write to files	<15>
Keyboard input	<6>	Search files	<16>
Display data	<7>	Equate file fields	<17>
Branches	<8>	Record check	<18>
Calculations	<9>	Clear files	<19>

From this menu, we'll select the options that will allow *TLO* to write our flowchart. *TLO* will then use this flowchart, and a few bits of other information, to write our final BASIC program.

Here we find such terms as "branch" and "pointers" and "keyboard input." Each step in the operation of our program will relate to the FCM in some manner. Expect to spend most of your PG time at or near this FCM menu.

As we proceed through the FCM process, the *TLO* keeps track of the program flowchart lines as each one is created. We have the opportunity to periodically review the flowchart to see where we are at any point in the FCM process. In the end, our completed flowchart looks like this:

1. Branch on menu: 1-3 to 2,7,18
2. Set pointer to end of mailing
3. Keyboard input using mailing fields
4. Write to mailing
5. Ask user "Any more ?":branch if no to 3
6. Direct unconditional branch to 1
7. Branch on menu: 1-3 to 8,13,1
8. Set pointer to start of mailing
9. Search mailing: If eof, branch to 11
10. Display data using mailing fields
11. Ask user "<Another Search ?>":branch if yes to 8
12. Direct unconditional branch to 1
13. Sort mailing
14. Set pointer to start of mailing
15. Read mailing: If eof, branch to 1
16. Display data using mailing fields
17. Direct unconditional branch to 15
18. Terminate program

The next step is called branch resolution. It's the process by which we advise the flowchart where to go from menus and from program decision points. This branch data is then inserted into the flowchart to complete that section of program creation.

At this point, the *TLO* offers us printed documentation and the ability to actually place our completed flowchart at the beginning of our BASIC program. Very handy when someone else has to read your program!

The final part of program creation takes the longest, but not really any more than any careful program user would care to spend. Here, we build our program menus and video displays, our search and display routines, and our printer report displays. (One feature we liked is the ability to center printed data on the screen and to draw graphic boxes on the videoscreen to highlight the menu or data we present.)

The data to appear on the screen is located by the respective row and column it will appear in. At any time, we have the ability to look at the actual screen or menu we are "building" and make additions or corrections at will. This section also allows us the flexibility to tell *TLO* if we want a data sort in our program and if so, how to sort it. In this manner, all important areas of the program are provided for.

The *TLO* then proceeds to write the completed program to the disk drive.

BUILDING UP MUSCLES

You might get the impression that creating your own program takes hours. It could, but most of the

program steps we described above really averaged only a few minutes each, with the longest section being the branch resolution sequence, requiring a little thinking and prior planning.

In fact, once you've "constructed" a few programs of your own, you will quickly come to appreciate how much work that program would've been if you had to write every line of your BASIC program by hand, without having a progam generator available!

The key in the operation of a program generator is not to jump directly into the PG expecting to write the ultimate user program. Take the time to experiment and begin slowly so that you can get into the flow. In fact, the best way to approach a PG is like one would a new car. Familiarize yourself with all the pedals and controls and you will enjoy the new machine sooner.

It was once said that "knowledge is power." With the arrival of programs that generate other programs, the modern-day computer user may well have the means by which his knowledge will unleash the true power of the personal computer. □

In the July issue of* Ahoy!, *the man who led you on a fantastic voyage inside the 1541 Disk Drive—Morton Kevelson—tears into the new MSD Super Disk Drive. (On sale June 12.)

ALPINER

Continued from page 47

Lines 190-220	Executes subroutines needed for program.
Lines 225-380	First screen loop.
Lines 390-560	Second screen loop.
Lines 570-750	Third screen loop.
Lines 800-895	Displays each run totals.
Lines 900-910	Loads assembly routine for joysticks.
Lines 1000-1150	Moves and changes character set.
Lines 1200-1280	Creates sprites.
Lines 1400-1460	Data for joystick routine.

If you prefer not to type the program in, send $5.00 and a stamped, self-addressed envelope to:

Bob Lloret-*Alpiner*
157 Atlantic Avenue
Staten Island, NY 10304

A copy will promptly be returned on disk or tape. (Please specify which your system requires!)

SEE PROGRAM LISTING ON PAGE 82

FLOTSAM

Finally, an alternative! Since acquiring your February issue I have eagerly awaited the March edition. After searching the contents page, I found the article that virtually made the price of a subscription jump from my pocket.

The reason I am so excited is the articles dealing with hardware. No other magazine that I'm aware of offers such indepth reviews with such great pictorial displays of the Commodore line of computers and peripherals.

The February and March editions, with articles like *The Anatomy of the Commodore 64,* have sold me on the future of this publication. How about a monthly column devoted to hardware, input/output, technical specifications and such?

Larry W. Odom
Homewood, AL

Thanks, Larry. If enough readers write to request such a column, we'll certainly implement it.

I was very impressed with your article on *Simons' BASIC* last month. In fact, it impressed me so much, I have purchased one, and am having a lot of fun learning to use the 114 additional commands it provides. In my opinion, it's an excellent addition for C-64 owners. But so far, I've noticed one quirk.

It seems that whenever the machine encounters a *Simons'* keyword in a data statement, it shortens it by replacing letters with graphic symbols. (Much the same way consol BASIC interprets keywords, i.e., the first letter and then a shifted second letter.)

It sort of took me by surprise the first time I used a data statement with the cartridge in place, but this only happens when you use one of the command names the extended version supports. Now for the solution. If the keyword is entered inside quotes, there seems to be no problem reading what's in the data statements and getting back to something legible. This isn't really an inconvenience unless you don't know about it. If you don't, it can be very frustrating trying to figure out why there is garbage on the screen. Just thought you folks should know.

Robert Radina
Moberly, MO

We welcome your comments on any aspect of Commodorean life. Write to Flotsam, *c/o* Ahoy!, *45 West 34th Street, Suite 407, New York, NY 10001.*

MEMORY MANAGEMENT

Continued from page 22

the Start of BASIC pointer. It then sets these first two bytes of the BASIC program to zero. The rest of the pointers are then reset in the same way as the CLR command. The result is that the BASIC interpreter cannot find either the original program or any of the variable data.

At this point, we can get down to business. The fundamental operation we will be doing is to modify the Start of BASIC pointer. Refer to the programs included with *Memory Management Part I* last issue. TEST I first identifies itself in line 30. The subroutine at line 200 just prints out the contents of the pointers. After waiting for a key to be hit in the subroutine at line 300, the first byte of the second program is set to zero by line 60. The Start of BASIC pointer is changed in line 70. For the VIC 20, the pointer is set to the 3K expansion area. Remember, both a 3K expander and an 8K expander must be installed in the VIC 20 for this demonstration. The start address of the 3K expansion area is 1024, as shown on the VIC 20 memory map last month. Remember that the Start of BASIC pointer must be set to 1025.

If this 3K area does not seem like much, then consider the following. The main program can load and pass control to a collection of small subroutines which are loaded into the 3K expander as required.

For the Commodore 64, the extra memory area is the 4K block starting at 49152. This RAM is built into the Commodore 64. Many machine language routines, such as the DOS wedge, are LOADed into this area. Just exercise some discretion when practicing these techniques.

Line 80 of Test I loads Test 2 into the new memory area specified by the new values POKEd into the Start of BASIC pointer. The actual transfer of control takes place when the LOAD is completed. Remember, when a LOAD is executed under program control, the BASIC interpreter will GOTO the start of the BASIC program as designated by the Start of BASIC pointer. Since we changed this to the new area before the LOAD was performed, Test 2 is the program which will be executed when the LOAD is completed.

Once the two programs are LOADed into their respective areas, we can still transfer control back and forth between them. This is demonstrated by line 70 which changes the Start of BASIC pointer and by the GOTO 10 statement in line 80. In this case, the line 10 being GOTOed is actually in Test 1. The way this is done makes use of the characteristics of the GOTO statement. If the high byte of the destination line number is greater than the high of the statement's line number, then the BASIC interpreter looks toward the end of the program for the destination. Otherwise, the start of the program is scanned. Since the Start of BASIC pointer was changed before the GOTO is executed, the scanning operation actually is performed on Test 1. This means that the line that is jumped should always have a lower line number than the line in which it appears. The best approach is to put the command line at the end of the program and the line which is jumped to at the beginning of the program.

Location 251, which is PEEKed at in line 20 of Test 1 and POKEd by line 60 of Test 2, is used as a flag by the programs. This lets Test 2 "tell" Test 1 it has been loaded in. This location is actually a Commodore freebie for machine language programmers. It is a zero page location which is not used by the operating system. This flag is used when transferring control back and forth between programs. It allows the calling program to instruct the called program to take different actions depending on results obtained by the calling program.

VIC 20 users will reap a big bonus if an 8K expansion module is installed in Block 5 which starts at 40960. This would be used as a third eight kilobyte chunk for these memory management techniques. Up to three different programs can reside in the VIC 20 memory map in this way. This is demonstrated by programs Test 3, Test 4, and Test 5 included in this issue.

When RUN, Tests 3, 4, and 5 will allow you to transfer control from any one to either of the others. Test 8 and Test 9 demonstrate the technique on the Commodore 64.

When writing the programs which will be loaded into these "new" memory locations, keep in mind that the available space is limited. For the VIC 20, a maximum program length of 3,071 bytes and 8,191 for 3K and 8K expanders can be entered and saved from the standard BASIC RAM. The following BASIC line will calculate the length of the program when executed:

```
PRINT (PEEK(46)-PEEK(44))*256+(PEE
K(45)-PEEK(43))
```

Until now, we have manipulated only the Start of BASIC pointer. It is possible to adjust the other

pointers as well. This would allow us to increase the space for variables into the expansion areas. It is a good idea to keep the variable space as a continuous section of RAM. The variable pointers are continuously adjusted by the BASIC interpreter as new variables are defined. It is also possible to readjust the pointers at the start of a program. The demonstration programs, Test 6 and Test 7, do this for the Commodore 64. In this case, Test 7 adjusts the pointers to assign the space used by Test 6 for variable storage. Programs, which are short compared to the data they manipulate, will benefit the most from this trick. Typical applications which fall into this category are word processors, mailing lists, databases, and disk copy routines.

We have now covered enough ground to get you started at Memory Management. A lot of interesting applications can be performed using these techniques. If you come up with some unusual ones, please send it to my attention care of *Ahoy!* Next time, you may see your own work in print!☐

SEE PROGRAM LISTINGS ON PAGE 80

RUPERT REPORT

Continued from page 62

by adding the first byte to 256 times the second byte. Line 20 gets the first byte from memory location 43 and assigns it to M1. Line 30 puts the second byte into M2. Line 40 then calculates the pointer's value and stores it in M.

Line 50 does the same thing in a more compact manner for the pointer to the start of BASIC variables. This pointer resides in addresses 45 and 46 as shown on page 312 of the *Programmer's Reference Guide*. Its value is stored in the variable N.

Since BASIC variables are stored just beyond program text, the end of our program must be at memory location (N-1). Line 60 sets up a FOR/NEXT loop that will step P through all memory locations of our program.

The value stored in each memory location is PEEKed at in line 70 and assigned to V. Line 80 changes the value of V to 166 if it is not in the range of 31 to 128. Line 90 displays the current memory address, then the value stored in that address, and finally the ASCII character corresponding to that value.

When the computer prints characters outside of the range of 31 to 128, peculiar things may happen on the screen. That is why any undesirable charac-

READER SERVICE INDEX

For information on any of the products listed here, circle the appropriate numbers on the Reader Service card between pages 90 and 91.

ters are changed to 166 in line 80. Where do these numbers come from? Look in the *ASCII and CHR$ Codes* appendix in your *Commodore User's Guide*. Character 166 is a checkered square. Character 147 would clear the screen and some other character would make the cursor invisible. Rather than risk that, we print a checkered square instead. Remove line 80 if you want to see for yourself.

Also, you will probably want to add a line such as this:

```
95 GET A$ : IF A$ = "" THEN 95
```

Hopefully, you now know what that line does. Hold down the space bar to continuously display memory locations. Release it when you want to study the screen.

We won't go too far into what all the strange things you see represent. That's the topic of another article. You will recognize all characters that are in REM statements. Variable names and numeric values are undisguised. If you look closely, you will see the line numbers.

On my Commodore 64, the program starts at location 2049. The first two bytes (22,8) are a pointer to the start of the next line in the program. The third and fourth bytes (10,0) represent the line number. Just as with pointers, multiply the second value by 256 then add it to the first value to calculate the line number. Here the result is 10.

The fifth byte (143) is a "token" or computer shorthand for the REM statement. Next comes the text in line 10. A zero value signifies the end of each line of the program. You should be able to evaluate the pointer from the first two bytes of the program to get the address for the start of line 20. Although your values may vary, my calculations give 2070 as the start of line 20 ($22+256*8=2070$). And sure enough, the zero at the end of line 10 is in location 2069!

The end of a program is signalled by three consecutive zeroes. With this program, you now have a tool to start some serious sleuthing into the innards of BASIC.

Well, there are still at least two other statements that are used to bring data into a BASIC program. They are related, and they both require more explanation than we have room for here. So next month we will look at the INPUT# and GET# statements. Do you have some data that you frequently want to update and analyze, but you are tired of typing and retyping it into the computer? These statements are just what you're looking for.

By the way, the answers to the quiz are 5, 10, 15, 20, 5, 0, 15, 1010. It was meant to be somewhat tricky. Congratulations if you got them all right.□

Next month in *The Rupert Report*, Dale does some investigation of sequential files in *The File Sleuth!*

POST TIME

Continued from page 35

call, the message "THEY'RE OFF!" will be displayed and the race will begin. The horses then move to the right of the screen or the finish line. In the 64 version, the first three horses to pass the ½ mile pole will be displayed at the bottom of the screen on their way to the finish line. The race ends when the first horse in the VIC version, or the first three in the 64 version, cross the finish line. The screen in the 64 version changes to a photo finish as the third horse crosses the line. In both versions, the results of the race will then be shown, with the players' names and the amounts won or lost. A brief pause, and the program goes automatically to the next race and displays the selections. The game continues until 10 races have been run.

If you prefer not to type the program in, do as follows:

C-64 VERSION: Send $5.00 and a stamped, self-addressed envelope to Bob Lloret-*Post Time,* 157 Atlantic Avenue, Staten Island, NY 10304. Specify whether you want your copy on disk or cassette!

VIC 20 VERSION: Send $3.00, a stamped, self-addressed envelope, and a blank disk or cassette to Nova Soft-*Post Time,* Box 527, Nutley, NJ 07112.

Hope you have as much fun playing *Post Time* as we've had. Pick a winner!

SEE PROGRAM LISTINGS ON PAGE 71

SOUND CONCEPT

Continued from page 40

To increase its portability, *Sounder* uses this area to store variables and will not tolerate outside alteration of the cells.

Boolean algebra is used to dense-pack the sound data bytes with information to be used by *Sounder*'s sound interpreter routine. Sound data are coded to represent characteristics of the sound to be produced and always come in a series of three. The first byte carries the selection of channel(s) for sound production, modifiers, and the volume. Byte 2 represents the sound's duration and may carry a strength value used by the "warble" sound modifier. The last of the data, byte 3, determines the sound's starting frequency. It also informs the sound processing routine if an additional sound is waiting to be initiated, and if so, processes the next 3 bytes of data.

USING SOUND CONCEPT IN OUTSIDE PROGRAMS

To use *Sound Concept* in a BASIC program, several small tasks need to be performed: *Sounder* must be placed in memory and protected from

Ahoy!

Back Issues



ISSUE #1—JAN. '84 $4.00
The 64 v. the Peanut! The computer as communications device! Protecto's Bill Badger interviewed! And ready to enter: the Multi Draw 64 graphics system! The Interrupt Music Maker/Editor! A Peek at Memory! Programming Sequential Files!

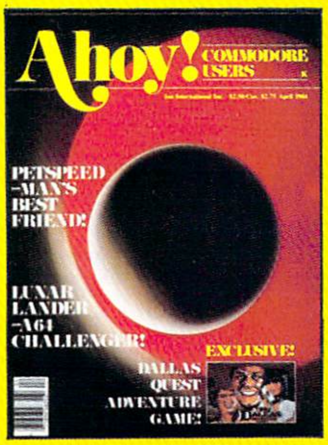
Don't punch another key without a complete collection of *Ahoy!* and the programming strategies and product analyses each issue provides. Order while supplies last!



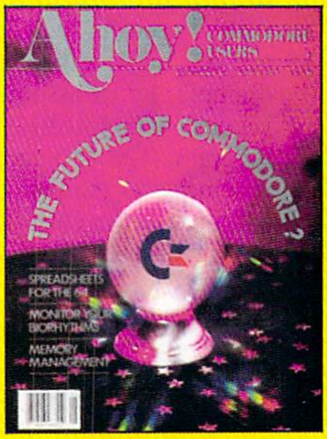
ISSUE #2—FEB. '84 $4.00
Illustrated tour of the 1541 disk drive! Synapse's Ihor Wolosenko interviewed! Users groups! Artificial intelligence! And ready to enter: Music Maker Part II! Night Attack! Programming Relative Files! Screen Manipulation on the Commodore 64!



ISSUE #3—MAR. '84 $4.00
Anatomy of the 64! Printer Interfacing for the 64 and VIC! Educational software: first of a series! Commodares! And ready to enter: Space Lanes! Random Files on the 64! Easy Access Address Book! Dynamic Power for your 64!



ISSUE #4—APR. '84 $4.00
Petspeed and Easy Script tutorials! Printer interfacing and educational software guide continued! Lower case descenders on your 1525! Laserdisc! The Dallas Quest Adventure Game! And ready to enter: Apple Pie! Lunar Lander! Name that Star!



ISSUE #5—MAY '84 $4.00
The Future of Commodore! Inside BASIC program storage! C-64 Spreadsheets! Memory Management on the VIC and 64! Educational Software Guide continues! And ready to enter: Math Master! Air Assault! Biorhythms! VIC 20 Calculator!

Send coupon or facsimile to:
Ahoy! Back Issues, Ion International Inc., 45 West 34th Street—Suite 407, New York, NY 10001

Ahoy!

Please Send Me The Following:
____ Copies of issue number ____
____ Copies of issue number ____
____ Copies of issue number ____
Enclosed Please Find My Check or Money Order for $________
(Outside the USA please add $1.00 for every copy)

NAME ________
ADDRESS ________
____
CITY ________
STATE ________
ZIP CODE ________

BASIC; a table of sound data must be established; and several lines of BASIC must be incorporated into the program.

Placing *Sounder* in memory and protecting it from BASIC is no problem. By following the same procedures discussed for using *Sounder* with the *Editor, Maker* will place and protect the machine language at the top of memory.

The option to combine *Sounder* with the BASIC program, through use of a machine language monitor to save the code, also applies. However, if you want to go this route, my advice is to use the top of memory as the place to keep *Sounder* during the development of the program. This way, it won't be wiped out when you alter the memory by changing and running BASIC. After program development is completed, the merge can be accomplished.

The sound data obtained from the *Editor* must also be placed in an area protected from BASIC. Again, the top of memory, below *Sounder,* can be used for this purpose if memory space permits and the quantity of data is rather large. However, the BASIC pointers to the top of memory, addresses 55 and 56, must be lowered to protect the data. As an example, let us assume that there are 150 individual sounds, with 450 bytes of data, to be used in the program. The 450 values will be included in data statements at the end of the program. To transfer to and protect the data at the top of memory, the following BASIC lines will do the job:

```
10 A = 256 * PEEK (56) + PEEK (55
) - 450:POKE 56,INT (A/256):POKE
55, A - 256 * PEEK * (56):CLR
20 A = 256 * PEEK (56) + PEEK (55
):FOR B = A TO A + 450:READ C:POK
E B,C:NEXT
```

Here, the line numbers, 10 and 20, are suggested and could be any legitimate line numbers. However, it is best to use these statements at the beginning of the program, as the CLR statement will wipe out any variables that precede it.

For most applications, I prefer to place sound data in the cassette buffer. As explained earlier, the first 16 bytes of the cassette, 838 to 846, are reserved for use by the machine language and can't be used to store sound data. However, 174 bytes remain free for this purpose, enough room for 58 individual sounds. To place the sound data in the cassette buffer, include the sound data in data statements at the end of the program, as above. Then, this single line can be used to make the transfer to the cassette buffer:

```
10 FOR A = 846 TO [ADDRESS OF LAS
T DATA VALUE]:READ B:POKE A, B:NE
XT
```

No other statements are required to protect the data.

Please note that when the cassette buffer is used to store the sound data, no tape files can be used in the program. If the top of memory is used for sound data storage, however, the program can use tape files if these procedures are followed: 1) turn off *Sounder* with a SYS statement; 2) use the cassette buffer for files; 3) clear addresses 828 and 829 with POKES of 0; and 4) turn *Sounder* back on with a SYS statement. Again, the addresses for turning *Sounder* on and off are provided by the *Maker* program.

The following program lines must also be incorporated into all outside BASIC programs that make use of *Sound Concept:*

```
30 POKE 828, 0:POKE 829, 0:SYS [A
DDRESS TO START SOUNDER PROVIDED
BY THE MAKER PROGRAM].
62000 POKE 1,INT (AD/256):POKE 0,
 AD -256 * PEEK (1):POKE 828,1:RE
TURN
```

The first series of statements, on Line 30, clears the memory cells that initiate sound. Therefore, when *Sounder* is turned on by the SYS statement, it does not go wild, using random values of memory as sound data.

The subroutine located on Line 62000, or whatever line number you choose to use, tells the machine language where the first byte of sound data for the desired sound/string is located and starts the sound.

To produce a desired sound within a program, all that is necessary is to include two statements at the point where the start of the sound is desired: AD =xxx:GOSUB 62000. The value of AD corresponds to the address of the first byte of sound data. That's all there is to it!

For those readers who are not willing to undertake the typing involved with *Sound Concept,* I will be glad to make copies of the programs. Send either a tape or disk, a self-addressed, stamped ($.50) mailer and $4.00 to:

A.J. Kwitowski
302 Euclid Ave.
Glassport, PA 15045

SEE PROGRAM LISTING ON PAGE 68

All the hits your computer is missing.



If you thought you'd never find fun games for your hardworking home computer, happy days are here. Because now ATARISOFT™ has all the great hits...Pac-Man[1] Donkey Kong[2] by Nintendo[2] Centipede™ Defender[3] Joust[3] Jungle Hunt[4] Moon Patrol[5] Pole Position[5] Galaxian[1] Ms. Pac-Man[1] and Battlezone™

And we've got them for all the hit computers ...Apple, IBM, Commodore 64, Vic-20, Colecovision,* and TI 99/4A. We've got Pac-Man, Centipede and Defender for Intellivision too.

So dust off your joystick and ask your dealer for all the ATARISOFT hits. It's the software your hardware's been waiting for.

ATARISOFT.™

All the hits your computer is missing.

Reader Service No. 265